* + - 1. General Coding unit syntax

|  |  |
| --- | --- |
| ~~if ( resPredEnableFlag )~~ |  |
| **~~res\_pred\_flag~~** | ~~ae(v)~~ |

* + - 1. General coding unit semantics

**~~res\_pred\_flag~~** ~~equal to 0 specifies that residual prediction is not used. res\_pred\_flag equal to 1 specifies that residual prediction is used. When res\_pred\_flag is not present, its value shall be inferred to be equal to 0.~~

~~[Ed. (GT). The following paragraph needs to be specified more precisely. ]~~

~~The variable residualCbfNonZero is derived by values of cbf\_luma, cbf\_cb, cbf\_cr and PredMode of the corresponding blocks as follows: residualCbfNonZero is set to 1 if at least one of the corresponding blocks has both PredMode not equal to MODE\_INTRA and any of the values of cbf\_luma, cbf\_cb and cbf\_cr not equal to 0; otherwise, residualCbfNonZero is set equal to 0. The corresponding blocks are identified by the current PU and the disparity vector. All the corresponding blocks belong to transform units that are covered or partially covered by a corresponding rectangle area (of the current PU) in the inter-view reference view component, after shifting the PU location with a disparity vector.~~

~~The variable resPredEnableFlag specifying whether res\_pred\_flag is present in the bitstream is derived as~~

* 1. ~~resPredEnableFlag = multi\_view\_residual\_pred\_flag && residualCbfNonZero && anyTempRefPicFlag (G‑)~~
     + 1. Derivation process for motion vector components and reference indices

…

Let ( xP, yP ) specify the top-left sample location of the current luma prediction block relative to the top-left luma sample of the current picture where xP = xC + xB and yP = yC + yB.

The flag ivRpFlag[xP][yP] is set to 0, and the disparity ivRpDisp[xP][yP] is set to 0.

Let the variable currPic and ListX be the current picture and RefPicListX (with X being 0 or 1) of the current picture, respectively.

…

* + - 1. Derivation process for luma motion vectors for merge mode

The motion vectors mvL0 and mvL1, the reference indices refIdxL0 and refIdxL1, ~~and~~ the prediction utilization flags predFlagL0 and predFlagL1, the disparity vector availability flags ivpMvFlagL0 and ivpMvFlagL1, and the disparity vectors ivpMvDispL0 and ivpMvDispL1 are derived as specified by the following ordered steps:

* 1. The derivation process for spatial merge candidates as specified in subclause G.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, the residual prediction flag ivRpFlagN, and the disparity used for residual prediction ivRpDispN 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 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.
     + 1. availableFlagCol = availableFlagL0Col | | availableFlagL1Col (G‑87)   
          predFlagLXCol = availableFlagLXCol (G‑88)
  4. Depending on multi\_view\_mv\_pred\_flag, the following applies.
     + If multi\_view\_mv\_pred\_flag is equal to 0, the flags availableFlagIvMC and availableFlagIvDC are set equal to 0.
     + Otherwise (multi\_view\_mv\_pred\_flag is equal to 1), the derivation process for the inter-view merge candidates as specified in subclause is invoked with the luma locations ( xC, yC ) and ( xP, yP ), the coding block size nCS, the variables nPSW and nPSH, and the partition index partIdx, as the inputs and the output is assigned to the view order index of the reference view refViewIdx, the availability flags availableFlagIvMC and availableFlagIvDC, the reference indices refIdxLXIvMC and refIdxLXIvDC, the prediction list utilization flags predFlagLXIvMC and predFlagLXIvDC, and the motion vectors mvLXIvMC and mvLXIvDC (with X being 0 or 1, respectively), and the disparity vector mvDisp, the residual prediction flag ivRpFlagIvMC, the disparity used for residual prediction ivRpDispIvMC.
  5. The merge candidate list, mergeCandList, is constructed as specified by the following ordered steps.
  6. The variable numMergeCand is set equal to 0.
  7. When availableFlagIvMC is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to IvMC and the variable numMergeCand is increased by 1.
  8. When availableFlagA1 is equal to 1 and one or more of the following conditions is true,
     + - * availableFlagIvMC  = = 0,
         * predFlagLXIvMC ! = predFlagLXA1, (with X being replaced by 0 and 1),
         * mvLXIvMC ! = mvLXA1 (with X being replaced by 0 and 1),
         * refIdxLXIvMC ! = refIdxLXA1 (with X being replaced by 0 and 1),

the entry mergeCandList[ numMergeCand ] is set equal to A1 and the variable numMergeCand is increased by 1.

* 1. When availableFlagB1 is equal to 1, and one or more of the following conditions is true ,
     + - * availableFlagIvMC = = 0,
         * predFlagLXIvMC ! = predFlagLXB1, (with X being replaced by 0 and 1),
         * mvLXIvMC ! = mvLXB1 (with X being replaced by 0 and 1),
         * refIdxLXIvMC ! = refIdxLXB1 (with X being replaced by 0 and 1),

the entry mergeCandList[ numMergeCand ] is set equal to B1 and the variable numMergeCand is increased by 1.

* 1. When availableFlagB0 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to B0 and the variable numMergeCand is increased by 1.
  2. When availableFlagIvDC is equal to 1, and one or more of the following conditions is true,
     + - * availableFlagA1 = = 0,
         * predFlagLXA1 ! = predFlagLXIvDC, (with X being replaced by 0 and 1),
         * mvLXA1 ! = mvLXIvDC(with X being replaced by 0 and 1),
         * refIdxLXA1 ! = refIdxLXIvDC(with X being replaced by 0 and 1),

and one or more of the following conditions is true,

* + - * + availableFlagB1 = = 0,
        + predFlagLXB1 ! = predFlagLXIvDC, (with X being replaced by 0 and 1),
        + mvLXB1 ! = mvLXIvDC(with X being replaced by 0 and 1),
        + refIdxLXB1 ! = refIdxLXIvDC(with X being replaced by 0 and 1),

the entry mergeCandList[ numMergeCand ] is set equal to IvDC and the variable numMergeCand is increased by 1.

* 1. When availableFlagA0 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to A0 and the variable numMergeCand is increased by 1.
  2. When availableFlagB2 is equal to 1 and numMergeCand is less than 4 + multi\_view\_mv\_pred\_flag, the entry mergeCandList[ numMergeCand ] is set equal to B2 and the variable numMergeCand is increased by 1.
  3. When availableFlagCol is equal to 1 and numMergeCand is less than 5 + multi\_view\_mv\_pred\_flag, the entry mergeCandList[ numMergeCand ] is set equal to Col and the variable numMergeCand is increased by 1.
  4. The variable numOrigMergeCand is set equal to numMergeCand.
  5. 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.
  6. 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.
  7. 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.
  8. The following assignments are made with N being the candidate at position MergeIdx ~~merge\_idx[ xP][ yP ]~~ in the merging candidate list mergeCandList ( N = mergeCandList[ MergeIdx ~~merge\_idx[ xP][ yP ]~~ ] ) and X being replaced by 0 or 1:
     + 1. mvLX[ 0 ] = mvLXN[ 0 ] (G‑89)
       2. mvLX[ 1 ] = mvLXN[ 1 ] (G‑90)
       3. refIdxLX = refIdxLXN (G‑91)
       4. predFlagLX = predFlagLXN (G‑92)
       5. when N is equal to IvMC, A0, A1, B0, B1 or B2, the fllowing applies:
          1. ivRpFlag[xP][yP] = ivRpFlagN
          2. ivRpDisp[xP][yP] = ivRpDispN
  9. When predFlagL0 is equal to 1 and predFlagL1 is equal to 1, and ( nPbW + nPbH ) is equal to 12, the following applies.
     + 1. refIdxL1 = −1 (G‑93)
       2. predFlagL1 = 0 (G‑94)
  10. The disparity availability flag ivpMvFlagLX and the disparity vector ivpMvDispLX are derived as follows (with X being replace by 0 or 1).
      + If all of the following conditions are true, ivpMvFlagLX is set equal to 1 and ivpMvDispLX is set equal to mvDisp [Ed. (GT) There is some redundancy in draft and software since ivpMvDispLX is derived for each list, although it is always equal for both lists.]
        - availableFlagIvMC = = 1
        - MergeIdx = = 0
        - predFlagLXIvMC = = 1

[Ed. (GT): PredMode[ xC ][ yC ] = = MODE\_SKIP might be added here instead of testing it in the disparity vector derivation process]

* + - Otherwise, ivpMvFlagLX is set equal to 0 and both components of ivpMvDispLX are set equal to 0.
      1. Derivation process for spatial merge candidates

The specifications in subclause 8.5.2.1.2 apply, with the following changes:

The following paragraph

* N is equal to B2 and availableFlagA0 + availableFlagA1 + availableFlagB0 + availableFlagB1 is equal to 4.

is removed.

The following paragraph (N being replaced by A0 , A1 , B0 , B1 , B2

* mvLXN = MvLX[ xN ][ yN ]
* refIdxLXN = RefIdxLX[ xN ][ yN ]
* predFlagLXN = PredFlagLX[ xN ][ yN ]

is replaced by

* mvLXN = MvLX[ xN ][ yN ]
* refIdxLXN = RefIdxLX[ xN ][ yN ]
* predFlagLXN = PredFlagLX[ xN ][ yN ]

if PartMode of current prediction unit is equal to PART\_2Nx2N, the following applies:

* + ivRpFlagN = ivRpFlag[xN][yN]
  + ivRpDispN = ivRpDisp[xN][yN]

The following outputs are added:

* the residual prediction flags ivRpFlagA0, ivRpFlagA1, ivRpFlagB0, ivRpFlagB1, ivRpFlagB2 of the neighbouring prediction units
* the disparity vectors used for residual prediction ivRpDispA0, ivRpDispA1, ivRpDispB0, ivRpDispB1, ivRpDispB2 of the neighbouring prediction units

The variables ivRpFlagN and ivRpDispN with N being replaced by A0 , A1 , B0 , B1 , B2  are initialized as 0.

* + - 1. Derivation process for inter-view merge candidates

…

Outputs of this process are (with X being 0 or 1, respectively)

* a view order index refViewIdx specifying a reference view.
* the availability flags availableFlagIvMC and availableFlagIvDC specifying whether the inter-view merge candidates are available,
* the reference indices refIdxLXIvMC and refIdxLXIvDC,
* the prediction list utilization flags predFlagLXIvMC and predFlagLXIvDC,
* the motion vectors mvLXIvMC and mvLXIvDC,
* the disparity vector mvDisp.
* a flag ivRpFlagIvMC specifying whether the inter-view residual prediction is used.
* the disparity vector ivRpDispIvMC specifying the disparity vector used for inter-view residual prediction

The derivation process for a disparity vector as specified in subclause G.8.5.2.1.13 is invoked with the luma locations ( xC, yC ) and ( xP, yP ), the coding block size nCS, the variables nPSW and nPSH, and the partition index partIdx, as the inputs and the outputs are and the view order index of the reference view refViewIdx, the flag availableDV and the disparity vector mvDisp.

The flag ivRpFlagIvMC is set 0, and the ivRpDispIvMC is set to 0.

The temporal inter-view motion vector merging candidate is derived by the following ordered steps.

* 1. For the prediction list indication X being 0 and 1 the following applies.
     + The derivation process for a temporal inter-view motion vector merging candidate as specified in subclause G.8.5.2.1.15 is invoked with the luma location ( xP, yP ), the variables nPSW and nPSH, the disparity vector mvDisp, the prediction list indication X and the view identifier refViewIdx as the inputs and the outputs are the flag availableFlagLXIvMC, the motion vector mvLXIvMC and the reference index refIdxLXIvMC.
  2. The availability flag availableFlagIvMC, and the prediction utilization flags predFlagL0IvMC and predFlagL1IvMC are derived by
     + 1. availableFlagIvMC = availableFlagL0IvMC | | availableFlagL1IvMC (G‑)  
          predFlagL0IvMC = availableFlagL0IvMC (G‑)  
          predFlagL1IvMC = availableFlagL1IvMC (G‑)
  3. When the availableFlagIvMC is equal to 1, the PredMode[xP,yP] is equal to SIZE\_2Nx2N, and the availableDV is equal to 1, the following applies.

ivRpFlagIvMC = 1

ivRpDispIvMC = mvDisp

The disparity inter-view motion vector merging candidate is derived by the following ordered steps.

…

* + - 1. Decoding process for inter prediction samples

…

* When ivRpFlag[xC][yC] is equal to 1, the inter-view residual prediction process as specified in subclause is invoked with the luma locations ( xC, yC ) and ( xB, yB ), the size of the current luma coding block nCS, the variables nPSW and nPSH, the reference view index refViewIdx set equal to 0, the prediction list utilization flags, predFlagL0 and predFlagL1, the reference indices refIdxL0 and refIdxL1, the luma motion vectors mvL0 and mvL1, chroma motion vectors mvCL0 and mvCL1, and the arrays predSamplesL, predSamplesCb, and predSamplesCr as the inputs and the outputs are modified versions of the arrays predSamplesL, predSamplesCb, and predSamplesCr.
  + - 1. Inter-view residual prediction process

Inputs to this process are:

* a luma location ( xC, yC ) specifying 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 luma sample of the current prediction unit relative to the top-left luma sample of the current picture,
* a variable nCS specifying the size of the current luma coding block,
* variables nPSW and nPSH specifying the width and the height, respectively, of the current prediction unit,prediction list utilization flags, predFlagL0 and predFlagL1,
* reference indices refIdxL0 and refIdxL1,
* luma motion vectors mvL0 and mvL1,
* chroma motion vectors mvCL0 and mvCL1,
* a (nPSW)x(nPSH) array predSamplesL of luma prediction samples,
* two (nPSW / 2)x(nPSH / 2) arrays predSamplesCb and predSamplesCr of chroma prediction samples.

Output of this process are:

* a variable refViewIdx specifying the view order index of the reference view disparity derivation,
* a modified version of the (nPSW)x(nPSH) array predSamplesL,
* a modified versions of the (nPSW / 2)x(nPSH / 2) arrays predSamplesCb and predSamplesCr.

[Ed. (GT). In software refViewIdx is set equal to the view order index of the view with minimal view\_id difference compared to the view\_id of the current view].

Set the disparity vector mvDisp[0] as ivRpDisp[xP][yP].

mvDisp[0] = (mvDisp[0]+2)>>2<<2

mvDisp[1] = 0

The following apply to derive the prediction values in the reference view:

* Set the reference picture sample arrays refPicL, refPicCb, and refPicCr corresponding to decoded sample arrays SL, SCb, SCr derived in subclause 8.7 for a previously-decoded picture which has the same POC value of current picture in the view component with ViewIdx equal to refViewIdx. The arrays refSamplesL, refSamplesCb, and refSamplesCr are derived by invoking the fractional sample interpolation process specified in subclause 8.5.3.2.2 with the luma locations ( xC, yC ), ( xB, yB ), the width and the height of the current luma prediction block nPbW, nPbH, the motion vectors mvDisp, mvDisp, and the reference arrays with refPicL, refPicCb andrefPicCr given as input.
* For X=0, and 1, the following applies.
  1. If the predFlagLX is equal to 1, the following applies.
  2. Set refPicLXL, refPicLXCb, and refPicLXCr corresponding to decoded sample arrays SL, SCb, SCr derived in subclause  for a previously-decoded picture which has the same POC value of RefPicListX[ refIdxLX ] in the view component with ViewIdx equal to refViewIdx. The arrays refPredSamplesLXL, refPredSamplesLXCb, and refPredSamplesLXCr are derived by invoking the fractional sample interpolation process specified in subclause 8.5.3.2.2 with the luma locations ( xC, yC ), ( xB, yB ), the width and the height of the current luma prediction block nPbW, nPbH, the motion vectors (mvLX[0]+ mvDisp[0], mvLX[1]+ mvDisp[1]), (mvLX[0]+ mvDisp[0], mvLX[1]+ mvDisp[1]) and the reference arrays with refPicLXL, refPicLXCb and refPicLXCr given as input.
* The array refPredSamplesL of the prediction samples of luma component is derived by invoking the weighted sample prediction process specified in subclause 8.5.3.2.3 with the luma location (xP, yP), the width and the height of the current luma prediction block nPbW, nPbH, and the sample arrays refPredSamplesL0L and. refPredSamplesL1L aswell as predFlagL0, predFlagL1, refIdxL0, refIdxL1 and cIdx equal to 0 given as input.
* The array refPredSamplesCb of the prediction samples of component Cb is derived by invoking the weighted sample prediction process specified in subclause 8.5.3.2.3 with the chroma location (xP/2, yP/2), the width and the height of the current chroma prediction block nPbW/2, nPbH/2, and the sample arrays refPredSamplesL0Cb and. refPredSamplesL1Cb aswell as predFlagL0, predFlagL1, refIdxL0, refIdxL1 and cIdx equal to 0 given as input.
* The array refPredSamplesCr of the prediction samples of component Cr is derived by invoking the weighted sample prediction process specified in subclause 8.5.3.2.3 with the chroma location (xP/2, yP/2), the width and the height of the current luma prediction block nPbW/2, nPbH/2, and the sample arrays refPredSamplesL0Cr and. refPredSamplesL1Cr aswell as predFlagL0, predFlagL1, refIdxL0, refIdxL1 and cIdx equal to 0 given as input.

The modified prediction samples predSamplesL[ x ][ y ] with x = 0..(nPbW)−1 and y = 0..(nPbH)−1 are derived as follows:

predSamplesL[ x ][ y ] = predSamplesL[ x ][ y ] + ( refSamplesL[ x ][ y ] - refPredSamplesL[ x ][ y ])

The modified prediction samples predSamplesCb[ x ][ y ], predSamplesCr[ x ][ y ] with x = 0..(nPbW / 2)−1 and y = 0..(nPbH / 2)−1 are derived as follows:

predSamplesCb[ x ][ y ] = predSamplesCb[ x ][ y ] + ( refSamplesCb[ x ][ y ] - refPredSamplesCb[ x ][ y ] )

predSamplesCr[ x ][ y ] = predSamplesCr[ x ][ y ] + ( refSamplesCr[ x ][ y ] - refPredSamplesCr[ x ][ y ] )