### 8.5.6 Decoding process for inter blocks

#### 8.5.6.1 General

This process is invoked when decoding a coding unit coded in inter prediction mode.

Inputs to this process are:

* a luma location ( xCb, yCb ) specifying the top-left sample of the current coding block relative to the top‑left luma sample of the current picture,
* a variable cbWidth specifying the width of the current coding block in luma samples,
* a variable cbHeight specifying the height of the current coding block in luma samples,
* variables numSbX and numSbY specifying the number of luma coding subblocks in horizontal and vertical direction,
* the motion vectors mvL0[ xSbIdx ][ ySbIdx ] and mvL1[ xSbIdx ][ ySbIdx ] with xSbIdx = 0 .. numSbX − 1, and ySbIdx = 0 .. numSbY − 1,
* the refined motion vectors refMvL0[ xSbIdx ][ ySbIdx ] and refMvL1[ xSbIdx ][ ySbIdx ] with xSbIdx = 0 .. numSbX − 1, and ySbIdx = 0 .. numSbY − 1,
* the reference indices refIdxL0 and refIdxL1,
* the prediction list utilization flags predFlagL0[ xSbIdx ][ ySbIdx ] and predFlagL1[ xSbIdx ][ ySbIdx ] with xSbIdx = 0 .. numSbX − 1, and ySbIdx = 0 .. numSbY − 1,
* the half sample interpolation filter index hpelIfIdx,
* the bi-prediction weight index bcwIdx,
* the mimimum sum of absolute difference values in decoder-side motion vector refimenent process dmvrSad[ xSbIdx ][ ySbIdx ] with xSbIdx = 0 .. numSbX − 1, and ySbIdx = 0 .. numSbY − 1,
* the decoder-side motion vector refinement flag dmvrFlag,
* a variable cIdx specifying the colour component index of the current block,
* the prediction refinement utilization flag cbProfFlagL0 and cbProfFlagL1,
* a motion vector difference array diffMvL0[ xIdx ][ yIdx ] and diffMvL1[ xIdx ][ yIdx ] with xIdx = 0..cbWidth / numSbX − 1, and yIdx = 0..cbHeight / numSbY − 1.

Outputs of this process are:

* an array predSamples of prediction samples.

Let predSamplesL0L, predSamplesL1L and predSamplesIntraL be (cbWidth)x(cbHeight) arrays of predicted luma sample values and, predSamplesL0Cb, predSamplesL1Cb, predSamplesL0Cr and predSamplesL1Cr, predSamplesIntraCb, and predSamplesIntraCr be (cbWidth / SubWidthC)x(cbHeight / SubHeightC) arrays of predicted chroma sample values.

* + The variable currPic specifies the current picture and the variable bdofFlag is derived as follows:
    - If all of the following conditions are true, bdofFlag is set equal to TRUE.
      * ph\_bdof\_disabled\_flag is equal to 0.
      * predFlagL0[ xSbIdx ][ ySbIdx ] and predFlagL1[ xSbIdx ][ ySbIdx ] are both equal to 1.
      * DiffPicOrderCnt( currPic, RefPicList[ 0 ][ refIdxL0 ] ) is equal to DiffPicOrderCnt( RefPicList[ 1 ][ refIdxL1 ], currPic).
      * RefPicList[ 0 ][ refIdxL0 ] is an STRP and RefPicList[ 1 ][ refIdxL1 ] is an STRP.
      * MotionModelIdc[ xCb ][ yCb ] is equal to 0.
      * merge\_subblock\_flag[ xCb ][ yCb ] is equal to 0.
      * sym\_mvd\_flag[ xCb ][ yCb ] is equal to 0.
      * ciip\_flag[ xCb ][ yCb ] is equal to 0.
      * BcwIdx[ xCb ][ yCb ] is equal to 0.
      * luma\_weight\_l0\_flag[ refIdxL0 ] and luma\_weight\_l1\_flag[ refIdxL1 ] are both equal to 0.
      * chroma\_weight\_l0\_flag[ refIdxL0 ] and chroma\_weight\_l1\_flag[ refIdxL1 ] are both equal to 0.
      * cbWidth is greater than or equal to 8.
      * cbHeight is greater than or equal to 8.
      * cbHeight \* cbWidth is greater than or equal to 128.
      * RprConstraintsActive[ 0 ][ refIdxL0 ] is equal to 0 and RprConstraintsActive[ 1 ][ refIdxL1 ] is equal to 0.
      * cIdx is equal to 0.
    - Otherwise, bdofFlag is set equal to FALSE.
* If numSbY is equal to 1 and numSbX is equal to 1 the following applies:
  + When bdofFlag is equal to TRUE, the variables numSbY, numSbX are modified as follows:

numSbX = ( cbWidth > 16 ) ? ( cbWidth  >>  4 ) : 1 (919)

numSbY = ( cbHeight > 16 ) ? ( cbHeight  >>  4 ) : 1 (920)

* + For X = 0..1, xSbIdx = 0..numSbX − 1 and ySbIdx = 0..numSbY − 1, the following applies:
    - predFlagLX[ xSbIdx ][ ySbIdx ] is set equal to predFlagLX[ 0 ][ 0 ].
    - refMvLX[ xSbIdx ][ ySbIdx ] is set equal to refMvLX[ 0 ][ 0 ].
    - mvLX[ xSbIdx ][ ySbIdx ] is set equal to mvLX[ 0 ][ 0 ].

The width and the height of the current coding sublock sbWidth, sbHeight in luma samples are derived as follows:

sbWidth  =  cbWidth / numSbX (921)

sbHeight  =  cbHeight / numSbY (922)

For each coding subblock at subblock index ( xSbIdx, ySbIdx ) with xSbIdx = 0 .. numSbX − 1, and ySbIdx = 0 .. numSbY − 1, the following applies:

* The luma location ( xSb, ySb ) specifying the top-left sample of the current coding subblock relative to the top‑left luma sample of the current picture is derived as follows:

( xSb, ySb )  =  ( xCb + xSbIdx \* sbWidth, yCb + ySbIdx \* sbHeight ) (923)

* For X being each of 0 and 1, when predFlagLX[ xSbIdx ][ ySbIdx ] is equal to 1, the following applies:
  + The reference picture consisting of an ordered two-dimensional array refPicLXL of luma samples and two ordered two-dimensional arrays refPicLXCb and refPicLXCr of chroma samples is derived by invoking the process specified in clause 8.5.6.2 with X and refIdxLX as inputs.
  + The motion vector offset mvOffset is set equal to refMvLX[ xSbIdx ][ xSbIdx ] − mvLX[ xSbIdx ][ ySbIdx ].
  + If cIdx is equal to 0, the following applies:
    - The array predSamplesLXL is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location ( xSb, ySb ), the coding subblock width sbWidth, the coding subblock height sbHeight in luma samples, the luma motion vector offset mvOffset, the refined luma motion vector refMvLX[ xSbIdx ][ xSbIdx ], the reference array refPicLXL, bdofFlag, dmvrFlag, hpelIfIdx, cIdx, RprConstraintsActive[ X ][ refIdxLX ], and RefPicScale[ X ][ refIdxLX ] as inputs.
    - When cbProfFlagLX is equal to 1, the prediction refinement with optical flow process specified in clause 8.5.6.4 is invoked with sbWidth, sbHeight, the (sbWidth + 2)x(sbHeight + 2) array predSamplesLXL and the motion vector difference array diffMvLX[ xIdx ][ yIdx ] with xIdx = 0..cbWidth / numSbX − 1, and yIdx = 0..cbHeight / numSbY − 1 as inputs and the refined (sbWidth)x(sbHeight) array predSamplesLXL as output.
  + Otherwise, if MotionModelIdc[ xCb ][ yCb ] is equal to 0, or the xSbIdx % SubWidthC and ySbIdx % SubHeightC are both equal to 0, the following applies:
    - ~~Otherwise, if~~ If cIdx is equal to 1, the following applies:
      * The array predSamplesLXCb is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location ( xSb, ySb ), the coding subblock width MotionModelIdc[ xCb ][ yCb ] ? sbWidth : sbWidth / SubWidthC, the coding subblock height MotionModelIdc[ xCb ][ yCb ] ? sbHeight  : sbHeight / SubHeightC, the chroma motion vector offset mvOffset, the refined chroma motion vector refMvLX[ xSbIdx ][ ySbIdx ], the reference array refPicLXCb, bdofFlag, dmvrFlag, hpelIfIdx, cIdx, RprConstraintsActive[ X ][ refIdxLX ], and RefPicScale[ X ][ refIdxLX ] as inputs.
    - Otherwise (cIdx is equal to 2), the following applies:
      * The array predSamplesLXCr is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location ( xSb, ySb ), the coding subblock width MotionModelIdc[ xCb ][ yCb ] ? sbWidth : sbWidth / SubWidthC, the coding subblock height MotionModelIdc[ xCb ][ yCb ] ? sbHeight  : sbHeight / SubHeightC, the chroma motion vector offset mvOffset, the refined chroma motion vector refMvLX[ xSbIdx ][ xSbIdx ], the reference array refPicLXCr, bdofFlag, dmvrFlag, hpelIfIdx, cIdx, RprConstraintsActive[ X ][ refIdxLX ], and RefPicScale[ X ][ refIdxLX ] as inputs.
* The variable sbBdofFlag is set equal to FALSE.
* When bdofFlag is equal to TRUE, the variable sbBdofFlag is further modifed as follows:
* If dmvrFlag is equal to 1 and the variable dmvrSad[ xSbIdx ][ ySbIdx ] is less than ( 2 \* sbWidth \* sbHeight ), the variable sbBdofFlag is set equal to FALSE.
* Otherwise, the variable sbBdofFlag is set equal to TRUE.
* The array predSamples of prediction samples is derived as follows:
* If cIdx is equal to 0, the prediction samples inside the current luma coding subblock, predSamples[ xL + xSb ][ yL + ySb ] with xL = 0..sbWidth − 1 and yL = 0..sbHeight − 1, are derived as follows:
* If sbBdofFlag is equal to TRUE, the bi-directional optical flow sample prediction process as specified in clause 8.5.6.5 is invoked with nCbW set equal to the luma coding subblock width sbWidth, nCbH set equal to the luma coding subblock height sbHeight and the sample arrays predSamplesL0L and predSamplesL1L, and the variables predFlagL0[ xSbIdx ][ ySbIdx ], predFlagL1[ xSbIdx ][ ySbIdx ], refIdxL0, and refIdxL1 as inputs, and predSamples[ xL + xSb ][ yL + ySb ] as outputs.
* Otherwise (sbBdofFlag is equal to FALSE), the weighted sample prediction process as specified in clause 8.5.6.6 is invoked with the luma coding subblock width sbWidth, the luma coding subblock height sbHeight and the sample arrays predSamplesL0L and predSamplesL1L, and the variables predFlagL0[ xSbIdx ][ ySbIdx ], predFlagL1[ xSbIdx ][ ySbIdx ], refIdxL0, refIdxL1, bcwIdx, dmvrFlag, and cIdx as inputs, and predSamples[ xL + xSb ][ yL + ySb ] as outputs.
* Otherwise, if cIdx is equal to 1, the prediction samples inside the current chroma component Cb coding subblock, predSamples[ xC + xSb / SubWidthC ][ yC + ySb / SubHeightC ] with xC = 0..sbWidth / SubWidthC − 1 and yC = 0..sbHeight / SubHeightC − 1, are derived by invoking the weighted sample prediction process specified in clause 8.5.6.6 with nCbW set equal to sbWidth / SubWidthC, nCbH set equal to sbHeight / SubHeightC, the sample arrays predSamplesL0Cb and predSamplesL1Cb, and the variables predFlagL0[ xSbIdx ][ ySbIdx ], predFlagL1[ xSbIdx ][ ySbIdx ], refIdxL0, refIdxL1, bcwIdx, dmvrFlag, and cIdx as inputs.
* Otherwise (cIdx is equal to 2), the prediction samples inside the current chroma component Cr coding subblock, predSamples[ xC + xSb / SubWidthC ][ yC + ySb / SubHeightC ] with xC = 0..sbWidth / SubWidthC − 1 and yC = 0..sbHeight / SubHeightC − 1, are derived by invoking the weighted sample prediction process specified in clause 8.5.6.6 with nCbW set equal to sbWidth / SubWidthC, nCbH set equal to sbHeight / SubHeightC, the sample arrays predSamplesL0Cr and predSamplesL1Cr, and the variables predFlagL0[ xSbIdx ][ ySbIdx ], predFlagL1[ xSbIdx ][ ySbIdx ], refIdxL0, refIdxL1, bcwIdx, dmvrFlag, and cIdx as inputs.
* When cIdx is equal to 0, the following assignments are made for x = 0..sbWidth − 1 and y = 0..sbHeight − 1:

MvL0[ xSb + x ][ ySb + y ] = mvL0[ xSbIdx ][ ySbIdx ] (924)

MvL1[ xSb + x ][ ySb + y ] = mvL1[ xSbIdx ][ ySbIdx ] (925)

MvDmvrL0[ xSb + x ][ ySb + y ] = refMvL0[ xSbIdx ][ ySbIdx ] (926)

MvDmvrL1[ xSb + x ][ ySb + y ] = refMvL1[ xSbIdx ][ ySbIdx ] (927)

RefIdxL0[ xSb + x ][ ySb + y ] = refIdxL0 (928)

RefIdxL1[ xSb + x ][ ySb + y ] = refIdxL1 (929)

PredFlagL0[ xSb + x ][ ySb + y ] = predFlagL0[ xSbIdx ][ ySbIdx ] (930)

PredFlagL1[ xSb + x ][ ySb + y ] = predFlagL1[ xSbIdx ][ ySbIdx ] (931)

HpelIfIdx[ xSb + x ][ ySb + y ] = hpelIfIdx (932)

BcwIdx[ xSb + x ][ ySb + y ] = bcwIdx (933)

When ciip\_flag[ xCb ][ yCb ] is equal to 1, the array predSamples of prediction samples is modified as follows:

* If cIdx is equal to 0, the following applies:
* The general intra sample prediction process as specified in clause 8.4.5.2.5 is invoked with the location ( xTbCmp, yTbCmp ) set equal to ( xCb, yCb ), the intra prediction mode predModeIntra set equal to INTRA\_PLANAR, the transform block width nTbW and height nTbH set equal to cbWidth and cbHeight, the coding block width nCbW and height nCbH set equal to cbWidth and cbHeight, and the variable cIdx as inputs, and the output is assigned to the (cbWidth)x(cbHeight) array predSamplesIntraL.
* The weighted sample prediction process for combined merge and intra prediction as specified in clause 8.5.6.7 is invoked with the location ( xTbCmp, yTbCmp ) set equal to ( xCb, yCb ), the coding block width cbWidth, the coding block height cbHeight, the sample arrays predSamplesInter and predSamplesIntra set equal to predSamples and predSamplesIntraL, respectively, and the colour component index cIdx as inputs, and the output is assigend to the (cbWidth)x(cbHeight) array predSamples.
* Otherwise, if cIdx is equal to 1 and cbWidth / SubWidthC is greater than or equal to 4, the following applies:
* The general intra sample prediction process as specified in clause 8.4.5.2.5 is invoked with the location ( xTbCmp, yTbCmp ) set equal to ( xCb / SubWidthC , yCb / SubHeightC ), the intra prediction mode predModeIntra set equal to INTRA\_PLANAR, the transform block width nTbW and height nTbH set equal to cbWidth / SubWidthC  and cbHeight / SubHeightC, the coding block width nCbW and height nCbH set equal to cbWidth / SubWidthC  and cbHeight / SubHeightC, and the variable cIdx as inputs, and the output is assigned to the (cbWidth / SubWidthC )x(cbHeight / SubHeightC) array predSamplesIntraCb.
* The weighted sample prediction process for combined merge and intra prediction as specified in clause 8.5.6.7 is invoked with the location ( xTbCmp, yTbCmp ) set equal to ( xCb, yCb ), the coding block width cbWidth / SubWidthC , the coding block height cbHeight / SubHeightC, the sample arrays predSamplesInter and predSamplesIntra set equal to predSamplesCb and predSamplesIntraCb, respectively, and the colour component index cIdx as inputs, and the output is assigend to the (cbWidth / SubWidthC )x(cbHeight / SubHeightC) array predSamples.
* Otherwise, if cIdx is equal to 2 and cbWidth / SubWidthC is greater than or equal to 4, the following applies:
* The general intra sample prediction process as specified in clause 8.4.5.2.5 is invoked with the location ( xTbCmp, yTbCmp ) set equal to ( xCb / SubWidthC , yCb / SubHeightC ), the intra prediction mode predModeIntra set equal to INTRA\_PLANAR, the transform block width nTbW and height nTbH set equal to cbWidth / SubWidthC  and cbHeight / SubHeightC, the coding block width nCbW and height nCbH set equal to cbWidth / SubWidthC  and cbHeight / SubHeightC, and the variable cIdx as inputs, and the output is assigned to the (cbWidth / SubWidthC )x(cbHeight / SubHeightC) array predSamplesIntraCr.
* The weighted sample prediction process for combined merge and intra prediction as specified in clause 8.5.6.7 is invoked with the location ( xTbCmp, yTbCmp ) set equal to ( xCb, yCb ), the coding block width cbWidth / SubWidthC , the coding block height cbHeight / SubHeightC, the sample arrays predSamplesInter and predSamplesIntra set equal to predSamplesCr and predSamplesIntraCr, respectively, and the colour component index cIdx as inputs, and the output is assigend to the (cbWidth / SubWidthC )x(cbHeight / SubHeightC) array predSamples.