G.8.5.2.1.1 Derivation process for luma motion vectors for merge mode

This process is only invoked when PredMode[ xC ][ yC ] is equal to MODE\_SKIP or PredMode[ xC ][ yC ] is equal to MODE\_INTER and merge\_flag [ xP ][ yP ] is equal to 1, where ( xP, yP ) specify the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture.

Inputs of this process are

* a luma location ( xC, yC ) of 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 sample of the current luma prediction block relative to the top-left luma sample of the current picture,
* a variable nCS specifying the size of the current luma coding block,
* variables specifying the width and the height of the luma prediction block, nPbW and nPbH,
* a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are

– the luma motion vectors mvL0 and mvL1,

– the reference indices refIdxL0 and refIdxL1,

– the prediction list utilization flags predFlagL0 and predFlagL1.

The variables singleMCLFlag is derived as follows.

* If log2\_parallel\_merge\_level\_minus2 is greater than 0 and nCS is equal to 8, singleMCLFlag is set to 1.
* Otherwise, singleMCLFlag is set to 0.

When singleMCLFlag is equal to 1, xP is set equal to xC, yP is set equal to yC, and both nPbW and nPbH are set equal to nCS.

NOTE – When singleMCLFlag is equal to 1, all the prediction units of the current coding unit share a single merge candidate list, which is identical to the merge candidate list of the 2Nx2N prediction unit.

The motion vectors mvL0 and mvL1, the reference indices refIdxL0 and refIdxL1, and the prediction utilization flags predFlagL0 and predFlagL1 are derived as specified by the following ordered steps:

1. Depending on multi\_view\_mv\_pred\_flag, the following applies.

– If multi\_view\_mv\_pred\_flag is equal to 0, the variable availableFlagInterView is set equal to 0.

– Otherwise (multi\_view\_mv\_pred\_flag is equal to 1), the derivation process for the inter-view merge candidate as specified in subclause is invoked with the luma location ( xP, yP ), the variables nPSW and nPSH, and the reference view identifier refViewIdx set equal to 0 as the inputs and the outputs are the flag availableFlagInterView, the prediction utilization flags predFlagL0InterView and predFlagL1InterView, the reference indices refIdxL0InterView and refIdxL1InterView, and the motion vectors mvL0InterView and mvL1InterView.

1. The derivation process for spatial merge candidates as specified in subclause 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 and refIdxL0InterView, refIdxL1InterView, mvL0InterView and mvL1InterView 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 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 8.5.2.1.7 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.

availableFlagCol = availableFlagL0Col | | availableFlagL1Col (G‑73)   
predFlagLXCol = availableFlagLXCol (G‑74)

1. ~~Depending on multi\_view\_mv\_pred\_flag, the following applies.~~

~~– If multi\_view\_mv\_pred\_flag is equal to 0, the variable availableFlagInterView is set equal to 0.~~

~~– Otherwise (multi\_view\_mv\_pred\_flag is equal to 1), the derivation process for the inter-view merge candidate as specified in subclause is invoked with the luma location ( xP, yP ), the variables nPSW and nPSH, and the reference view identifier refViewIdx set equal to 0 as the inputs and the outputs are the flag availableFlagInterView, the prediction utilization flags predFlagL0InterView and predFlagL1InterView, the reference indices refIdxL0InterView and refIdxL1InterView, and the motion vectors mvL0InterView and mvL1InterView.~~

1. The merge candidate list, mergeCandList, is constructed as specified by the following ordered steps:.
   1. The variable numMergeCand is set equal to 0.
   2. When availableFlagInterView is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to InterView and the variable numMergeCand is increased by 1.
   3. When availableFlagA1 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to A1 and the variable numMergeCand is increased by 1.
   4. When availableFlagB1 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to B1 and the variable numMergeCand is increased by 1.
   5. When availableFlagB0 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to B0 and the variable numMergeCand is increased by 1.
   6. When availableFlagA0 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to A0 and the variable numMergeCand is increased by 1.
   7. When availableFlagB2 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to B2 and the variable numMergeCand is increased by 1.
   8. When availableFlagCol is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to Col and the variable numMergeCand is increased by 1.
2. The variable numOrigMergeCand is set equal to numMergeCand .
3. 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.
4. 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.
5. 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.

1. The following assignments are made with N being the candidate at position MergeIdx in the merging candidate list mergeCandList ( N = mergeCandList[ MergeIdx ] ) and X being replaced by 0 or 1:

mvLX[ 0 ] = mvLXN[ 0 ] (G‑75)

mvLX[ 1 ] = mvLXN[ 1 ] (G‑76)

refIdxLX = refIdxLXN (G‑77)

predFlagLX = predFlagLXN (G‑78)

1. When predFlagL0 is equal to 1 and predFlagL1 is equal to 1, and ( nPbW + nPbH ) is equal to 12, the following applies.

refIdxL1 = −1 (G‑79)

predFlagL1 = 0 (G‑80)

1. When the MergeIdx is not equal to 0, the following applies.

IvpMvFlagL0[ xP , yP ] = 0 (G‑81)

IvpMvFlagL1[ xP , yP ] = 0 (G‑82)

**8.5.2.1.2 Derivation process for spatial merging candidates**

Inputs to this process are

* a luma location ( xC, yC ) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
* a variable nCS specifying the size of the current luma coding block,
* a luma location ( xP, yP ) specifying the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
* a variable singleMCLFlag,
* variables specifying the width and the height of the luma prediction block, nPbW and nPbH,
* a variable partIdx specifying the index of the current prediction unit within the current coding unit.
* refIdxL0InterView, refIdxL1InterView, mvL0InterView and mvL1InterView

Outputs of this process are (with N being replaced by A0, A1, B0, B1 or B2 and with X being replaced by 0 or 1)

* the availability flags availableFlagN of the neighbouring prediction units,
* the reference indices refIdxLXN of the neighbouring prediction units,
* the prediction list utilization flags predFlagLXN of the neighbouring prediction units,
* the motion vectors mvLXN of the neighbouring prediction units.

For the derivation of availableFlagN, with N being A0, A1, B0, B1 or B2 and ( xN, yN ) being ( xP − 1,  yP + nPbH ), ( xP − 1,  yP + nPbH − 1 ), ( xP + nPbW,  yP − 1 ), ( xP + nPbW − 1,  yP − 1 ) or ( xP − 1,  yP − 1 ), the following applies.

– When yP−1 is less than (( yC >> Log2CtbSizeY ) << Log2CtbSizeY), the following applies.

xB0 = ((xB0>>3)<<3) + ((xB0>>3)&1)\*7 (8‑80)  
xB1 = ((xB1>>3)<<3) + ((xB1>>3)&1)\*7 (8‑81)  
xB2 = ((xB2>>3)<<3) + ((xB2>>3)&1)\*7 (8‑82)

– The availability derivation process for a prediction block as specified in subclause 6.4.2 is invoked with the luma location ( xC, yC ), the current luma coding block size nCbS set equal to nCS, the luma location ( xP, yP ), the width and the height of the luma prediction block nPbW and nPbH, the luma location ( xN, yN ) and the partition index partIdx as inputs and the output is assigned to the prediction block availability flag availableN.

– If one or more of the following conditions are true with X being replaced by 0 and 1, the availableFlagN is set equal to 0, both components mvLXN are set equal to 0, refIdxLXN is set equal to −1 and predFlagLXN is set equal to 0.

* (xP >> (log2\_parallel\_merge\_level\_minus2 + 2)) is equal to (xN >> (log2\_parallel\_merge\_level\_minus2 + 2)) and (yP >> (log2\_parallel\_merge\_level\_minus2 + 2)) is equal to (yN >> (log2\_parallel\_merge\_level\_minus2 + 2)).
* N is equal to B2 and availableFlagA0 + availableFlagA1 + availableFlagB0 + availableFlagB1 is equal to 4.
* availableN is equal to FALSE
* singleMCLFlag is equal to 0 and PartMode of the current prediction unit is PART\_2NxN or PART\_2NxnU or PART\_2NxnD and partIdx is equal to 1 and N is equal to B1
* singleMCLFlag is equal to 0 and PartMode of the current prediction unit is PART\_Nx2N or PART\_nLx2N or PART\_nRx2N and partIdx is equal to 1 and N is equal to A1
* N is equal to A1 and mvL0InterView is equal to mvL0N and refIdxL0InterView is equal to refIdxL0N and mvL1InterView is equal to mvL1N and refIdxL1InterView is equal to refIdxL1N
* N is equal to B1 and mvL0InterView is equal to mvL0N and refIdxL0InterView is equal to refIdxL0N and mvL1InterView is equal to mvL1N and refIdxL1InterView is equal to refIdxL1N
* N is equal to B1 and the prediction units covering luma location ( xA1, yA1 ) and luma location ( xN, yN ) have the same motion vectors and the same reference indices
* N is equal to B0 and the prediction units covering luma location ( xB1, yB1 ) and luma location ( xN, yN ) have the same motion vectors and the same reference indices
* N is equal to A0 and the prediction units covering luma location ( xA1, yA1 ) and luma location ( xN, yN ) have the same motion vectors and the same reference indices
* N is equal to B2 and the prediction units covering luma location ( xA1, yA1 ) and luma location ( xN, yN ) have the same motion vectors and the same reference indices
* N is equal to B2 and the prediction units covering luma location ( xA1, yA1 ) and luma location ( xN, yN ) have the same motion vectors and the same reference indices

– Otherwise, availableFlagN is set equal to 1 and the following assignments are made.

mvLXN = MvLX[ xN ][ yN ] (8‑83)  
refIdxLXN = RefIdxLX[ xN][ yN ] (8‑84)  
predFlagLXN = PredFlagLX[ xN ][ yN ] (8‑85)