The changes for DDD are highlighted in yellow.

The changes for the second texture candidate are highlighted in green.

H.7.4.7.1

…

**cp\_scale**[ j ], **cp\_off**[ j ], **cp\_inv\_scale\_plus\_scale**[ j ], and **cp\_inv\_off\_plus\_off**[ j ]specify conversion parameters for converting a depth value to a disparity value. When not present, the values of cp\_scale[ j ], cp\_off[ j ], cp\_inv\_scale\_plus\_scale[ j ], and cp\_inv\_off\_plus\_off[ j ], are inferred to be equal to vps\_cp\_scale[ ViewIdx ][ j ], vps\_cp\_off[ ViewIdx ][ j ], vps\_cp\_inv\_scale\_plus\_scale[ ViewIdx ][ j ], and vps\_cp\_inv\_off\_plus\_off[ ViewIdx ][ j ], respectively. It is a requirement of bitstream conformance, that the values of cp\_scale[ j ], cp\_off[ j ], cp\_inv\_scale\_plus\_scale[ j ], and cp\_inv\_off\_plus\_off[ j ] in a slice segment header having a ViewIdx equal to viewIdxA and the values of cp\_scale[ j ], cp\_off[ j ], cp\_inv\_scale\_plus\_scale[ j ], and cp\_inv\_off\_plus\_off[ j ] in a slice segment header having a ViewIdx equal to viewIdxB shall be the same, when viewIdxA is equal to viewIdxB.

The variables DDDInvScale[j], DDDShift[j] and DDDInvOffset[j] are derived as follows.

The variable AbsScale is set equal to abs(cp\_scale[ j ]).

The variable CpWidth is set equal to ⌊log2(cp\_scale[ j ])⌋+9.

The variable TargetV is set equal to 1<< CpWidth.

The variable BestD is set equal to 256.

The variable MinError is set equal to abs(AbsScale \*256- TargetV).

For d from 256 to 511, the following applies.

If abs(AbsScale \*d- TargetV) < MinError, MinError is set equal to abs(AbsScale \*d- TargetV), and BestD is set equal to d.

If TargetV -AbsScale \* BestD is equal to 0, the variable RoundingDir is set equal to 0. Otherwise, RoundingDir is set equal Sign(TargetV -AbsScale \* BestD).

DDDInvScale[j] is set equal to (BestD<<(BitDepthY +cp\_precision-1 ))\*Sign(cp\_scale[ j ]).

DDDShift[i] is set equal to CpWidth.

DDDInvOffset[i] is set equal to

-Sign(cp\_scale[ j ])\* BestD\*(cp\_off[ j ]<< BitDepthY)+ (1<<( DDDShift[i]-1)) + (1<<( DDDShift[i]-4))\* RoundingDir.

…

H.8.5.3.2.1

…

Outputs of this process are

…

* the flag subPbMotionFlag, specifying, whether the motion data of the current PU has sub prediction block size motion accuracy,
* the flag DDDModeFlag, specifying whether the current PU is coded using disparity derived depth,
* the value disp2depthValue, specifying the disparity derived depth value for the current PU.

…

1. Depending on mpi\_flag[ nuh\_layer\_id ], the following applies
   * + mergeCandIsDDDFlag[i] is set equal to 0 for i from 0 to 5 + iv\_mv\_pred\_flag [ nuh\_layer\_id ] + DepthFlag.
     + If mpi\_flag[ nuh\_layer\_id ] is equal to 0, the variable availableFlagT is set equal to 0.
     + Otherwise (mpi\_flag[ nuh\_layer\_id ] is equal to 1), the variables OffsetW\_C and OffsetH\_C are set equal to ( nPSW – 1 ) >> 1 and ( nPSH – 1 ) >> 1 respectively. variables OffsetW\_RB and OffsetH\_RB are set equal to nPSW and nPSH respectively. For N being replaced by C and RB, the derivation process for the texture merging candidate as specified in subclause H.8.5.3.2.14 is invoked with the luma location ( xPb, yPb ), the variables nPbW and nPbH, the offsets OffsetW\_N and OffsetH\_N as the inputs and the outputs are the flag availableFlagTN, the flag availableFlagDN, the disparity derived depth value disp2depthN, the prediction utilization flags predFlagL0TN and predFlagL1TN, predFlagL0DN and predFlagL1DN, the reference indices refIdxL0TN and refIdxL1TN, and refIdxL0DN, refIdxL1DN, and the motion vectors mvL0TN and mvL1TN and mvL0DN and mvL1DN.
2. The merging candidate list, extMergeCandList, is constructed as follows:

i = 0

if( availableFlagTC )

extMergeCandList[ i++ ] = TC

if( availableFlagDC )

extMergeCandList[ i++ ] = DC

…

if( availableFlagB2 )

extMergeCandList[ i++ ] = B2

if( availableFlagTRB && i < ( 5 + NumExtraMergeCand && differentMotion( TRB, extMergeCandList[0]) )

extMergeCandList[ i++ ] = TRB

if( availableFlagIvMCShift && i < ( 5 + NumExtraMergeCand ) && ( !availableFlagIvMC  | |  differentMotion( IvMC, IvMCShift ) ) )

extMergeCandList[ i++ ] = IvMCShift

…

16. The variable DDDModeFlag is set equal to (N== DC). When DDDModeFlag is equal to 1, the variable disp2depthValue is set equal to disp2depthC.

H.8.5.3.2.14

…

Inputs to this process are:

* a luma location ( xPb, yPb ) of the top-left luma sample of the current prediction unit relative to the top-left luma sample of the current picture,
* variables nPbW and nPbH specifying the width and the height, respectively, of the current prediction unit,
* variables nOffsetX and nOffsetY specifying the offset position to get the texture location.

Outputs of this process are:

…

* the motion vectors mvL0T and mvL1T (when availableFlagT is equal to 1),
* a flag availableFlagD specifying whether the disparity derived candidate is available,
* the prediction utilization flags predFlagL0D and predFlagL1D,
* the reference indices refIdxL0D and refIdxL1D (when availableFlagD is equal to 1),
* the motion vectors mvL0D and mvL1D (when availableFlagD is equal to 1),
* the disparity derived depth value disp2depth.
* The variable availableFlagT is set equal to 0. The variables predFlagL0T and predFlagL1T are set equal to 0. The variables refIdxL0T and refIdxL1T are set equal to −1. Both components of the motion vectors mvL0T and mvL1T are set equal to 0.
* The variable availableFlagD is set equal to 0. The variables predFlagL0D and predFlagL1D are set equal to 0. The variables refIdxL0D and refIdxL1D are set equal to −1. Both components of the motion vectors mvL0D and mvL1D are set equal to 0. disp2depth is set equal to 0.

…

The texture luma location ( xRef, yRef ) is derived by:

xRefFull = xPb + nOffsetX

yRefFull = yPb + nOffsetY

…

1. When X is equal to 0 or the current slice is a B slice, the following applies:
   * + When textPredFlagLX[ xRef ][ yRef ] is equal to 1, the following applies:
       1. mvLXT[ 0 ] = ( textMvLX[ xRef ][ yRef ][ 0 ] + 2 ) >> 2
       2. mvLXT[ 1 ] = ( textMvLX[ xRef ][ yRef ][ 1 ] + 2 ) >> 2
       3. refIdxLX = textRefIdxLX[ xRef ][ yRef ]
       4. predFlagLXT = 1
       5. availableFlagT = 1
          1. When PicOrderCnt( RefPicListX[ refIdxLX] ) is equal to PicOrderCnt of the current picture and availableFlagD is equal to 0, the following applies
          2. mvLXD[ 0 ] = mvLXT[ 0 ]
          3. mvLXD[ 1 ] = mvLXT[ 1 ]
          4. refIdxLXD = refIdxLX
          5. predFlagLXD = 1
          6. availableFlagD = 1
          7. ViewId = ViewId( RefPicListX[ refIdxLX] ), DV= textMvLX[ xRef ][ yRef ][ 0 ]
          8. disp2depthTmp =
          9. (DDDInvScale[ViewId] \* DV + DDDInvOffset[ViewId])>> DDDShift[ViewId]
          10. disp2depth = Clip3( 0, ( 1 << bitDepth ) − 1, disp2depthTmp)

8.5.3.3 Decoding process for inter prediction samples

…

The variable nCSL is set equal to nCS and the variable nCSC is set equal to nCS >> 1.

* If DDDModeFlag[ xC + xB ][ yC + yB ] is equal to 1, the following applies:

For each luma sample location ( xL = 0..nPbW−1, yL = 0..nPbH−1 ) inside the prediction luma sample array predSamplesL, the corresponding prediction luma sample value predSamplesL[ xL ][ yL ] is set equal to disp2depthValue[ xC + xB ][ yC + yB ]

Else

…