I.8.4.4.2.9 Depth partition value derivation and assignment process

Inputs to this process are:

* the neighbouring samples p[ x ][ y ], with x = −1, y = −1..nTbS \* 2 − 1 and x = 0..nTbS \* 2 − 1, y = −1,
* a binary array partitionPattern[ x ][ y ], with x, y =0..nTbS − 1, specifying a partitioning of the prediction block in a partition 0 and a partition 1,
* a sample location ( xTb, yTb ) specifying the top-left sample of the current block relative to the top‑left sample of the current picture,
* a variable nTbS specifying the transform block size.

Output of this process is:

* the predicted samples predSamples[ x ][ y ], with x, y = 0..nTbS − 1.

The variables vertEdgeFlag and horEdgeFlag are derived as specified in the following:

* 1. vertEdgeFlag = ( partitionPattern[ 0 ][ 0 ]  !=  partitionPattern[ nTbS − 1 ][ 0 ] ) (I‑64)
  2. horEdgeFlag = ( partitionPattern[ 0 ][ 0 ]  !=  partitionPattern[ 0 ][ nTbS − 1 ] ) (I‑65)

The variables dcValBR and dcValLT are derived as specified in the following:

* If vertEdgeFlag is equal to horEdgeFlag, the following applies:
  + The variable dcValBR is derived as follows:
    - If horEdgeFlag is equal to 1, the following applies:
      * 1. dcValBR = ( ( p[ −1 ][ nTbS − 1 ] + p[ nTbS − 1 ][ −1 ] ) >> 1 ) (I‑66)
    - Otherwise (horEdgeFlag is equal to 0), the following applies:
      * 1. vertAbsDiff = Abs( p[ −1 ][ 0 ] − p[ −1 ][ nTbS \* 2 − 1 ] ) (I‑67)
        2. horAbsDiff = Abs( p[ 0 ][ −1 ] − p[ nTbS \* 2 − 1 ][ −1 ] ) (I‑68)
        3. dcValBR = ( horAbsDiff > vertAbsDiff ) ? p[ nTbS \* 2 − 1 ][ −1 ] : p[ −1 ][ nTbS\*2 − 1 ] ) (I‑69)
  + The variable dcValLT is derived as follows:
    1. dcValLT = ( p[ −1 ][ 0 ] + p[ 0 ][ −1 ] ) >> 1 (I‑70)
* Otherwise (horEdgeFlag is not equal to vertEdgeFlag), the following applies:
  + 1. dcValBR = horEdgeFlag ? p[ −1 ][ nTbS − 1 ] : p[ nTbS − 1 ][ −1 ] (I‑71)
    2. dcValLT = horEdgeFlag ? p[ ( nTbS − 1 ) >> 1 ][ −1 ] : p[ −1 ][ ( nTbS − 1 ) >> 1 ] (I‑72)

The flag dcOffsetAvailFlag is set equal to ( !sdc\_flag[ xTb ][ yTb ]  &&  depth\_dc\_flag[ xTb ][ yTb ] ).

1. The predicted sample values predSamples[ x ][ y ] are derived as specified in the following:

* For x in the range of 0 to ( nTbS − 1 ), inclusive the following applies:
  + For y in the range of 0 to ( nTbS − 1 ), inclusive the following applies:
    - The variables predDcVal and dcOffset are derived as specified in the following:
      * 1. predDcVal = ( partitionPattern[ x ][ y ] = = partitionPattern[ 0 ][ 0 ] ) ? dcValLT : dcValBR (I‑73)
        2. dcOffset = dcOffsetAvailFlag ? DcOffset[ xTb ][ yTb ][ partitionPattern[ x ][ y ] ] : 0 (I‑74)
    - If DltFlag[ nuh\_layer\_id ] is equal to 0, the following applies:
      * 1. predSamples[ x ][ y ] = predDcVal + dcOffset (I‑75)
    - Otherwise (DltFlag[ nuh\_layer\_id ] is equal to 1), the following applies:
      * 1. predSamples[ x ][ y ] = Idx2DepthValue[ DepthValue2Idx[ predDcVal ] + dcOffset ] (I‑76)

1. The variables dcVal0 and dcVal1 are derived as specified in the following:
   * + If DltFlag[ nuh\_layer\_id ] is equal to 0, the following applies:
2. dcVal0 = dcValLT + dcOffsetAvailFlag ? DcOffset[ xTb ][ yTb ][0] : 0
3. dcVal1 = dcValBR + dcOffsetAvailFlag ? DcOffset[ xTb ][ yTb ][1] : 0
   * + Otherwise (DltFlag[ nuh\_layer\_id ] is equal to 1), the following applies:

dcVal0 = Idx2DepthValue [ DepthValue2Idx[ dcValLT ] + dcOffsetAvailFlag ? DcOffset[ xTb ][ yTb ][0] : 0 ]

dcVal1 = Idx2DepthValue [ DepthValue2Idx[ dcValBR ] + dcOffsetAvailFlag ? DcOffset[ xTb ][ yTb ][1] : 0 ]

I8.4.4.3 Segmental depth intra coding process

Inputs to this process are:

* a luma location ( xTb, yTb ) specifying the top-left luma sample of the current block relative to the top-left luma sample of the current picture,
* a variable nTbS specifying the transform block size,
* the intra prediction mode predModeIntra,

Output of this process is:

* reconstructed depth value samples resSamples[ x ][ y ], with x, y = 0.. nTbS − 1.

1. The array of predicted samples predSamples[ x ][ y ], with x, y =0..nTbS − 1, is derived as follows:

* The variable nPartTbS is set equal to Min( nTbS, 1  <<  Log2MaxTrafoSize ).
* The variable numPartsInRow is set equal to ( nTbS / nPartTbS ).
* For i in the range of 0 to ( numPartsInRow \* numPartsInRow − 1), inclusive, the following applies:
  + The variable xOffset is set to nPartTbS \* ( i % numPartsInRow ).
  + The variable yOffset is set to nPartTbS \* ( i / numPartsInRow ).
    1. [ Ed. (GT): Raster scan order instead of z-scan order is used here. Input from the proponent to fix this appreciated. ]
  + The general intra sample prediction process as specified in subclause 8.4.4.2.1 is invoked with the transform block location ( yTb0 + xOffset , yTb0 + yOffset ), the intra prediction mode predModeIntra, the transform block size nPartTbS, and the variable cIdx as inputs, and the output is an (nPartTbS)x(nPartTbS) array predSamplesPart.
    1. [ Ed. (GT): It seems not to be clear which are the neighbouring samples, when above process is invoked. HEVC version 1 text is in general not so specific about this. However, here things get worse, since for some neighbouring blocks only the prediction signal (without SDC residual) is available at this point. whereas for other the reconstructed values are available. ]
  + For x, y = 0..nPartTbS − 1, inclusive, predSamples[ x + xOffset ][ y + yOffset ] is set equal to predSamplesPart[ x ][ y ].

Depending on predModeIntra the array wedgePattern[ x ][ y ] with x, y =0..nTbS − 1 specifying the binary segmentation pattern is derived as follows:

* If predModeIntra is equal to INTRA\_DMM\_WFULL, the following applies:
  + 1. wedgePattern = WedgePatternTable[ Log2( nTbS ) ][ wedge\_full\_tab\_idx[ xTb ][ yTb ] ]
* Otherwise, if predModeIntra is equal to INTRA\_DMM\_CPREDTEX, the derivation process for a texture predicted contour pattern as specified in subclause I.8.4.4.4 is invoked with the sample location ( xTb, yTb ), and the variable nTbS as inputs, and the output is the binary partition pattern wedgePattern[ x ][ y ], with x, y =0..nTbS − 1.
* Otherwise (predModeIntra is not equal to INTRA\_DMM\_WFULL or INTRA\_DMM\_CPREDTEX), the following applies:
  + For x, y = 0..nTbS − 1 wedgePattern[ x ][ y ] is set equal to 0.

Depending on DltFlag[ nuh\_layer\_id ] the reconstructed depth value samples resSamples[ x ][ y ] are derived as specified in the following:

* If DltFlag[ nuh\_layer\_id ] is equal to 0, the following applies:
  + For x, y = 0..nTbS − 1, the reconstructed depth value samples resSamples[ x ][ y ] are derived as specified in the following:
    - 1. SL[ xTb0 + x ][ yTb0 + y ] = predSamples[ x ][ y ] + DcOffset[ xTb ][ yTb ][wedgePattern[ x ][ y ] ] (I‑78)
* Otherwise (DltFlag[ nuh\_layer\_id ] is equal to 1), the following applies:
  + The variables dcPred[ 0 ] and dcPred[ 1 ] are derived as specified in the following:
    - If predModeIntra is not equal to INTRA\_DMM\_WFULL or INTRA\_DMM\_CPREDTEX, the following applies:
      * 1. dcPred[ 0 ] = ( predSamples[ 0 ][ 0 ] + predSamples[ 0 ][ nTbS − 1 ] + predSamples[ nTbS − 1 ][ 0 ]  
            + predSamples[ nTbS − 1 ][ nTbS − 1 ] + 2 ) >> 2 (I‑79)

Otherwise the following applies:dcPred[0] is set as dvVal0 as derived in subclause I.8.4.4.2.9

dcPred[1] is set as dvVal1 as derived in subclause I.8.4.4.2.9

* + For x, y = 0..nTbS − 1, the reconstructed depth value samples resSamples[ x ][ y ] are derived as specified in the following:
    - 1. dltIdxPred = DepthValue2Idx[ dcPred[ wedgePattern[ x ][ y ] ] ] (I‑85)
      2. dltIdxResi = DcOffset[ xTb ][ yTb ][wedgePattern[ x ][ y ] ] (I‑86)
      3. SL[ xTb0 + x ][ yTb0 + y ] = predSamples[ x ][ y ] +   
          Idx2DepthValue[ dltIdxPred + dltIdxResi ] − dcPred[ wedgePattern[ x ][ y ] ] (I‑87)