#### Derivation process for ALF transpose and filter index for luma samples

Inputs of this process are:

* a luma location ( xCtb, yCtb ) specifying the top-left sample of the current luma coding tree block relative to the top left sample of the current picture,
* a reconstructed luma picture sample array recPictureL prior to the adaptive loop filtering process.

Outputs of this process are

* the classification filter index array filtIdx[ x ][ y ] with x, y = 0..CtbSizeY − 1,
* the transpose index array transposeIdx[ x ][ y ] with x, y = 0..CtbSizeY − 1.

The variables ac[ x ][ y ], sumH[ x ][ y ], sumV[ x ][ y ], sumD0[ x ][ y ], sumD1[ x ][ y ] and sumOfHV[ x ][ y ] with x, y = 0..( CtbSizeY − 1 ) >> 2 are derived as follows:

* + The variables x4 and y4 are set as ( x << 2 ) and ( y << 2 ), respectively.
* The variables applyAlfLineBufBoundary, minY, maxY, and ac[ x ][ y ] are derived as follows:
* If y4 is equal to ( CtbSizeY − 8 ) and one of the following condition is true, minY is set equal to −2, maxY is set equal to 3, and ac[ x ][ y ] is set equal to 3.
  + The bottom boundary of the current coding tree block is the bottom boundary of the picture and pic\_height\_in\_luma\_samples − yCtb > CtbSizeY − 4.
  + The bottom boundary of the current coding tree block is not the bottom boudary of the picture.
* Otherwise, if y4 is equal to ( CtbSizeY − 4 ) and one of the following condition is true, minY is set equal to 0, maxY is set equal to 5, and ac[ x ][ y ] is set equal to 3.
  + The bottom boundary of the current coding tree block is the bottom boundary of the picture and pic\_height\_in\_luma\_samples − yCtb > CtbSizeY − 4.
  + The bottom boundary of the current coding tree block is not the bottom boudary of the picture
* Otherwise, minY is set equal to −2, maxY is set equal to 5, and ac[ x ][ y ] is set equal to 2.
  + The variables clipLeftPos, clipRightPos, clipTopPos, clipBottomPos, clipTopLeftFlag and clipBotRightFlag are derived by invoking the ALF boundary position derivation process as specified in clause 8.8.5.5 with ( xCtb, yCtb ) and ( x4, y4 ) as inputs.
  + The locations ( hx4 + i, vy4 + j ) for each of the corresponding luma samples inside the given array recPicture of luma samples with i, j = −3..6 are derived as follows:

hx4 + i = Clip3( 0, pic\_width\_in\_luma\_samples − 1, xCtb + x4 + i ) (1427)

vy4 + j = Clip3( 0, pic\_height\_in\_luma\_samples − 1, yCtb + y4 + j ) (1428)

* The variables hx4 + i and vy4 + j are modified by invoking the ALF sample padding process as specified in clause 8.8.5.6 with ( xCtb, yCtb ), ( hx4 + i, vy4 + j ), the variable isChroma set equal to 0, clipLeftPos, clipRightPos, clipTopPos, clipBottomPos, clipTopLeftFlag and clipBotRightFlag as input.
  + The variables filtH[ i ][ j ], filtV[ i ][ j ], filtD0[ i ][ j ] and filtD1[ i ][ j ] with i, j = − 2..5 are derived as follows:
  + If both i and j are even numbers or both i and j are not even numbers, the following applies:

filtH[ i ][ j ] = Abs( ( recPicture[ hx4 + i, vy4 + j ]  <<  1 ) − recPicture[ hx4 +  i − 1, vy4 +j ] −  (1429)  
  recPicture[ hx4 + i + 1, vy4 + j ] )

filtV[ i ][ j ] = Abs( ( recPicture[ hx4 + i, vy4 + j ]  <<  1 ) − recPicture[ hx4 + i, vy4 + j − 1 ] −  (1430)  
  recPicture[ hx4 + i, vy4 + j + 1 ] )

filtD0[ i ][ j ] = Abs( ( recPicture[ hx4 + i, vy4 + j ]  <<  1 ) − recPicture[ hx4 + i − 1, vy4 + j − 1 ] −  (1431)  
 recPicture[ hx4 + i + 1, vy4 + j + 1 ] )

filtD1[ i ][ j ] = Abs( ( recPicture[ hx4 + i, vy4 + j ]  <<  1 ) − recPicture[ hx4 + i + 1, vy4 + j − 1 ] −  (1432)  
 recPicture[ hx4 + i − 1, vy4 + j + 1 ] )

* + Otherwise, filtH[ i ][ j ], filtV[ i ][ j ], filtD0[ i ][ j ] and filtD1[ i ][ j ] are set equal to 0.
  + The variables sumH[ x ][ y ], sumV[ x ][ y ], sumD0[ x ][ y ], sumD1[ x ][ y ] and sumOfHV[ x ][ y ] are derived as follows:

sumH[ x ][ y ] = ΣiΣj filtH[ i ][ j ], with i = − 2..5, j = minY..maxY (1433)

sumV[ x ][ y ] = ΣiΣj filtV[ i ][ j ], with i = − 2..5, j = minY..maxY (1434)

sumD0[ x ][ y ] = ΣiΣj filtD0[ i ][ j ], with i = − 2..5, j = minY..maxY (1435)

sumD1[ x ][ y ] = ΣiΣj filtD1[ i ][ j ], with i = − 2..5, j = minY..maxY (1436)

sumOfHV[ x ][ y ] = sumH[ x ][ y ] + sumV[ x ][ y ] (1437)

The classification filter index array filtIdx and transpose index array transposeIdx are derived by the following steps:

1. The variables dir1[ x ][ y ], dir2[ x ][ y ] and dirS[ x ][ y ] with x, y = 0..CtbSizeY − 1 are derived as follows:

* The variables hv1, hv0 and dirHV are derived as follows:
* If sumV[ x >> 2 ][ y >> 2 ] is greater than sumH[ x >> 2 ][ y >> 2 ], the following applies:

hv1 = sumV[ x >> 2 ][ y >> 2 ] (1438)

hv0 = sumH[ x >> 2 ][ y >> 2 ]  (1439)

dirHV = 1 (1440)

* Otherwise, the following applies:

hv1 = sumH[ x >> 2 ][ y >> 2 ] (1441)

hv0 = sumV[ x >> 2 ][ y >> 2 ] (1442)

dirHV = 3 (1443)

* The variables d1, d0 and dirD are derived as follows:
* If sumD0[ x >> 2 ][ y >> 2 ] is greater than sumD1[ x >> 2 ][ y >> 2 ], the following applies:

d1 = sumD0[ x >> 2 ][ y >> 2 ] (1444)

d0 = sumD1[ x >> 2 ][ y >> 2 ] (1445)

dirD = 0 (1446)

* Otherwise, the following applies:

d1 = sumD1[ x >> 2 ][ y >> 2 ] (1447)

d0 = sumD0[ x >> 2 ][ y >> 2 ] (1448)

dirD = 2 (1449)

* The variables hvd1, hvd0, are derived as follows:

hvd1 = ( d1 \* hv0 > hv1 \* d0 )  ?  d1  :  hv1 (1450)

hvd0 = ( d1 \* hv0 > hv1 \* d0 )  ?  d0  :  hv0 (1451)

* The variables dirS[ x ][ y ], dir1[ x ][ y ] and dir2[ x ][ y ] derived as follows:

dir1[ x ][ y ] = ( d1 \* hv0 > hv1 \* d0 )  ?  dirD  :  dirHV (1452)

dir2[ x ][ y ] = ( d1 \* hv0 > hv1 \* d0 )  ?  dirHV  :  dirD (1453)

dirS[ x ][ y ] = ( hvd1 \*2 > 9 \* hvd0 )  ?  2  :  ( ( hvd1 > 2 \* hvd0 )  ?  1  :  0 ) (1454)

1. The variable avgVar[ x ][ y ] with x, y = 0..CtbSizeY − 1 is derived as follows:

varTab[ ] = { 0, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 4 } (1455)

avgVar[ x ][ y ] = varTab[ Clip3( 0, 15, (1456)  
 ( sumOfHV[ x >> 2 ][ y >> 2 ] \* ac[ x >> 2 ][ y >> 2 ] ) >> ( BitDepth – 1 ) ) ]

1. The classification filter index array filtIdx[ x ][ y ] and the transpose index array transposeIdx[ x ][ y ] with x = y = 0..CtbSizeY − 1 are derived as follows:

transposeTable[ ] = { 0, 1, 0, 2, 2, 3, 1, 3 }

transposeIdx[ x ][ y ] = transposeTable[ dir1[ x ][ y ] \* 2 + ( dir2[ x ][ y ] >> 1 ) ]

filtIdx[ x ][ y ] = avgVar[ x ][ y ]

* When dirS[ x ][ y ] is not equal 0, filtIdx[ x ][ y ] is modified as follows:

filtIdx[ x ][ y ] += ( ( ( dir1[ x ][ y ] & 0x1 ) << 1 ) + dirS[ x ][ y ] ) \* 5 (1457)