#### Filtering process for luma samples

Inputs of this process are:

– a luma location ( xC, yC ) specifying the top-left luma sample of the current coding unit relative to the top left luma sample of the current picture,

– a variable log2CUSize specifying the size of the current coding unit,

– a filter index array of (nS)x(nS), fIdx.

Output of this process is the filtered reconstruction of luma picture.

The boundary padding process specified in subclause 8.6.3.1 is invoked with the luma location ( xC, yC ), the size of coding unit log2CUSize and the chroma component index cIdx set equal to 0, and the output is assigned to the luma sample array s’’. [Ed. (WJ): s’’ is now a picture-size array, but actually CU size + appropriate margin is enough]

A variable nS is set equal to ( 1 << log2CUSize ) and a variable alfFilterShape is set equal to alf\_filter\_shape\_flag.

A variable lcuHeight is set equal to ( 1 << Log2MaxCUSize ) and a variable vbLine is set equal to lcuHeight – 4.

Each sample of luma picture recFiltPictureL[ xC + x ][ yC + y ] with x, y = 0..(nS)-1, is derived as following ordered steps:

1. A variable dist2VB is derived as follows.

dist2VB = ( ( yC + y ) % lcuHeight – vbLine ) (8‑476)

1. A variable dist2VB is modified as follows.

* If dist2VB is less than –vbLine+2 and yC is larger than 2, dist2VB is set equal to dist2VB+lcuHeight,
* Otherwise, if yC+lcuHeight is equal or larger than pic\_height\_in\_luma\_samples, dist2VB is set equal to 5.

1. If alfFilterShape is equal to 0 and dist2VB is equal to 0 or -1, the following applies.

 (8‑476)

1. The following applies.

 (8‑476)

Table 8‑18 – Specification of horPos[ i ] according to alfFilterShape for adaptive loop filter process

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **i** | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| dist2VB < -2 or dist2VB > 1 | alfFilterShape = 0 | -2 | 0 | 2 | -1 | 0 | 1 | -2 | -1 | 0 | 1 | 2 | -1 | 0 | 1 | -2 | 0 | 2 |
| dist2VB == -2  or dist2VB == 1 | alfFilterShape = 0 | 0 | 0 | 0 | -1 | 0 | 1 | -2 | -1 | 0 | 1 | 2 | -1 | 0 | 1 | 0 | 0 | 0 |
| dist2VB == -1or dist2VB == 0 | alfFilterShape = 0 | 0 | 0 | 0 | 0 | 0 | 0 | -2 | -1 | 0 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| alfFilterShape = 1 | | 0 | 0 | 0 | 0 | -4 | -3 | -2 | -1 | 0 | 1 | 2 | 3 | 4 | 0 | 0 | 0 | 0 |

Table 8‑19 – Specification of verPos[ i ] according to alfFilterShape for adaptive loop filter process

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **i** | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| dist2VB < -2 or dist2VB > 1 | alfShape == 0 | -2 | -2 | -2 | -1 | -1 | -1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 2 | 2 | 2 |
| dist2VB == -2 | alfShape == 0 | -2 | -2 | -2 | -1 | -1 | -1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| dist2VB == -1 | alfShape == 0 | 0 | 0 | 0 | -1 | -1 | -1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| dist2VB == 0 | alfShape == 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| dist2VB == 1 | alfShape == 0 | -1 | -1 | -1 | -1 | -1 | -1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 2 | 2 | 2 |
| dist2VB < -4 or dist2VB > 3 | alfShape == 1 | -4 | -3 | -2 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 3 | 4 |
| dist2VB == -4 or dist2VB == 3 | alfShape == 1 | 0 | -3 | -2 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 3 | 0 |
| dist2VB == -3 or dist2VB == 2 | alfShape == 1 | 0 | 0 | -2 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 |
| dist2VB == -2 or dist2VB == 1 | alfShape == 1 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| dist2VB == -1 or dist2VB == 0 | alfShape == 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



(a) alfFilterShape == 0 (b) alfFilterShape == 1

Figure 8‑6 Mapping between geometric position and luma adaptive loop filter index according to alfFilterShape (informative)

#### Filtering process for chroma samples

Inputs of this process are:

– a chroma location ( xC, yC ) specifying the top-left luma sample of the current coding unit relative to the top left chroma sample of the current picture,

– a variable log2CUSize specifying the size of the current coding unit.

– a variable cIdx specifying the chroma component index.

Output of this process is the filtered reconstruction of chroma picture.

The boundary padding process specified in subclause 8.6.3.1 is invoked with the chroma location ( xC, yC ), the size of coding unit log2CUSize and the chroma component index cIdx, and the output is assigned to the luma sample array s’’. [Ed. (WJ): s’’ is now a picture-size array, but actually CU size + appropriate margin is enough]

A variable nS is set equal to ( 1 << log2CUSize ) and a variable alfFilterShape is set equal to alf\_filter\_shape\_chroma\_flag.

A variable lcuHeight is set equal to ( 1 << ( Log2MaxCUSize – 1 ) ) and a variable vbLine is set equal to lcuHeight – 2.

Filtered samples of chroma picture recFiltPicture[ xC + x ][ yC + y ] with x, y = 0..(nS)-1, are derived as the following ordered steps:

1. A variable dist2VB is derived as follows.

dist2VB = ( ( yC + y ) % lcuHeight – vbLine ) (8‑476)

1. A variable dist2VB is modified as follows.

* If dist2VB is less than –vbLine+2 and yC is larger than 2, dist2VB is set equal to dist2VB+lcuHeight,
* Otherwise, if yC+lcuHeight is equal or larger than pic\_height\_in\_luma\_samples >> 1, dist2VB is set equal to 5.

1. The following applies.

 (8‑476)