

Apparatus and Method for Filtering in Video Coding

TECHNICAL FIELD

[0001] Generally, the present invention relates to the field of video coding. More specifically, the present invention relates to a filter for video coding, method for filtering reconstructed video frames, and method for filtering video blocks as well as an encoding apparatus and a decoding apparatus comprising such the filter for video coding.

BACKGROUND

[0002] Digital video has been widely used since the introduction of DVD-discs. Before transmission the video is encoded and transmitted using a transmission medium. The viewer receives the video and uses a viewing device to decode and display the video. Over the years the quality of video has improved, for example, because of higher resolutions, color depths and frame rates. This has lead into larger

[0034] FIG. 8 shows an example of combining results of four 2x2 groups filtering with reusing of results of same spatial group final filter;

[0035] FIG. 9 as an example, shows the result of optimizing the LUT;

[0036] FIG. 10 as an example, shows undesirable gap in filter transfer function;

[0037] FIG. 11 as an example, represents same table with table entries being plot one-by-one;

[0038] FIG. 12 as an example, illustrates the method how gap can be eliminated using auxiliary function;

[0039] FIG. 13 illustrates the example of eliminating gap by taken maximum from two values while generating the LUT;

[0040] FIG. 14, as an example, illustrates filter transfer function after applying described above method;

[0041] FIG. 15 illustrates example of filter transfer functions depending on five QPs in the set;

[0042] FIG. 16 illustrates example of filter transfer functions for five QPs in the set based on corresponding tables;

[0043] FIG. 17 illustrates example of filter transfer functions for five QPs in the set based on corresponding tables; and

[0044] FIG. 18 is a schematic diagram illustrating an exemplary structure of an apparatus according to an embodiment.

[0045] In the various figures, identical reference signs will be used for identical or functionally equivalent features.

DETAILED DESCRIPTION OF EMBODIMENTS

[0046] In the following description, reference is made to the accompanying drawings, which form part of the disclosure, and in which are shown, by way of illustration, specific aspects in which the present invention may be placed. It is understood that other aspects may be utilized and structural or logical changes may be made without departing from the scope of the present invention. The following detailed description, therefore, is not to be taken in a limiting sense, as the scope of the present invention is defined by the appended claims.

[0047] For instance, it is understood that a disclosure in connection with a described method may also hold true for a corresponding device or system configured

decoding apparatus 200. A reconstruction unit 214 of the decoding apparatus 200 is configured to reconstruct the block on the basis of the filtered predicted block and the residual block provided by the inverse quantization unit 210 and the inverse transformation unit 212. As in the case of the encoding apparatus 100, the reconstructed block can be provided to a line buffer 216 used for intra prediction and the filtered block/frame can be provided to a decoded picture buffer 230 by the filter 220 for inter prediction.

[0056] As shown in Fig. 2B, the prediction signal in the hybrid decoder embodiment shown in Fig. 2B obtain from intra or inter prediction is further filtered by filter 264.

[0057] As already described above, the filter 120, 220 may be used at a frame level, for example, the filter 120, 220 may be configured to process a reconstructed frame from a decoded reconstructed video stream for generating a filtered reconstructed frame, where the reconstructed frame includes a plurality of blocks. The filter 120, 220 may be also used at a block level immediately after block reconstruction without waiting for a whole frame, for example, the filter 120, 220 may be configured to process a reconstructed block for generating a filtered reconstructed block, where the reconstructed block includes a plurality of pixels.

[0058] The filter 120, 220, 145 or 264 comprises one or more processor (or one or more processing unit). As will be explained in more detail below, the one or more processor (or one or more processing unit) is configured to: load a current pixel and its neighboring pixels to a linear buffer according to a predefined scan template (in other words, scan order, or scan pattern); obtain spectrum components by performing 1D transform for each pixel in the linear buffer; obtain filtered spectrum components by multiplying each spectrum component with a gain coefficient, wherein the gain coefficient depends on a corresponding spectrum component and a filtering parameter; obtain filtered pixels by performing inverse 1D transform for filtered spectrum; and generate a filtered reconstructed block based on the filtered pixels estimated on previous processing steps. In an example, the gain coefficient depends on a corresponding spectrum component and a filtering parameter. In another example, the gain coefficient depends on one or more filtering parameters and one or more corresponding spectrum components. In other example, the respective gain coefficient may depend on one or more filtering parameters and the corresponding spectrum

component as well as neighboring spectral components to the left and to the right of the spectrum component.

[0059] The disclosure describes in-loop filter for lossy video codec which performs local and/or non-local filtering of reconstructed block from reconstructed frame. According to an example, the reconstructed frame is divided into set of small non-overlapped rectangular blocks (CU blocks). On the next step each reconstructed block (reconstructed CU block) is filtered in frequency domain independently from other reconstructed blocks. The filter can also be applied after transform and reconstruction, and the filtered result is used both for output as well as for spatial and temporal prediction.

[0060] As another example, the disclosure describes prediction filter for lossy video codec which performs local and/or non-local filtering of prediction block of reconstructed frame.

[0061] At the first step of processing all pixels inside reconstructed block can be processed independently from each other. For processing of pixel $r(0)$, neighboring pixels are used. For example, as illustrated on Figure 3A, pixels $r(1)$ to $r(7)$ are used, and pixels $r(0)$ to $r(7)$ form one processing group.

[0062] Fig. 3A or 3B shows a schematic diagram 300, 300' illustrating aspects of a filtering process implemented in a filter according to an embodiment. At step 302, 302', a current pixel and its neighboring pixels from a block are loaded to a linear buffer according to a predefined scan template. As an example, the block may be a predicted block. As another example, the block may be a reconstructed block.

[0063] At step 304, 304', 1D transform is performed for pixel $r(0)$ and its neighboring pixels $r(1)$ to $r(7)$ in the linear buffer to obtain spectrum components R :

$$R = 1D_Transform(r)$$

[0064] As an example, the 1D transform may be a Hadamard transform.

[0065] It should be understood that whether to perform 1D transform on 4 pixels in the row (e.g. pixels A,B,C,D as in example on Fig. 3B) or perform 2D transform to spatially located pixels A,B,C,D relates to implementation specific. Applying 2D transform on 4 pixels A,B,C,D located in 2x2 block can lead to same result as applying 1D transform to 4 pixels A,B,C,D being taken as a row. At step 306, 306', filtering is performed in frequency domain based on multiplication of each spectrum component

[00154] In some implementation keeping LUT size limited and having THR in order to keep of equation from para [0077] maybe contradictory. Indeed at high QP values (which leads to high σ value) keeping LUT size limited by e.g. 16 entries may lead to undesirable gap in filter transfer function (which is represented by LUT) around value 120 as depicted on FIG. 10 (which also includes method of LUT subsampling by dropping of 3 least significant bits off).

[00155] FIG. 11 represents same table with table entries being plot one-by-one without consideration LUT subsampling effect illustrating gap between last LUT entry at index 15 and next point in filter transfer function at index 16.

[00156] FIG. 12 illustrates the method how gap can be eliminated using auxiliary function coming through THR value at the argument of corresponding to last LUT entry + 1, e.g. by using straight line equation (pointed by green color), coming through point THR=128 at argument 16 that corresponds to argument of last LUT entry (which is 15) increased by 1 or in other words LUT size (which is 16) and crossing x-axis at some point (e.g. at a value of 11 like in given example). It should be noted that other types of auxiliary functions can be used based on same principle, e.g. exponential, logarithmic, parabolic etc. or combination of them.

[00157] FIG. 13 Illustrates the example of eliminating gap by taken maximum from two values while generating the LUT, where the first value is LUT entry as described above $LUT(R_i, \sigma) = \frac{R_i^3}{R_i^2 + m * \sigma^2}$; and the second is a value of AuxiliaryFunction (straight line in this example) at the same argument i:

$$LUT(R_i, \sigma) = \max\left(\frac{R_i^3}{R_i^2 + m * \sigma^2}, AuxiliaryFunc_{\sigma}(i)\right);$$

where $AuxiliaryFunc_{\sigma}(i)$ represents the auxiliary function, and the auxiliary function has a value equal to THR at i corresponding to an argument of last LUT + 1.

[00158] FIG. 14 illustrates filter transfer function after applying described above method of using auxiliary straight line equation and LUT subsampling by dropping 3 least significant bits.

[00159] As described in para [00144] one LUT can used for group of QPs. To cover the possible QP range the predefined QP set is used and one LUT is generated for each QP from the set. The FIG.15 illustrates example of filter transfer functions depending on five QPs in the set and corresponding tables (from 0 to 4). In this

example the table 4 is generated using method described in paras [00156]-[00157] with straight-line auxiliary function crossing x-axis at the value of 11; and table 3 is generated using same method with straight-line auxiliary function crossing x-axis at the value of 9. The tables used in this example have values according to:

table0 = { 0, 2, 10, 19, 28, 36, 45, 53, 61, 70, 78, 86, 94, 102, 110, 118, },

table1 = { 0, 0, 5, 12, 20, 29, 38, 47, 56, 65, 73, 82, 90, 98, 107, 115, },

table2 = { 0, 0, 1, 4, 9, 16, 24, 32, 41, 50, 59, 68, 77, 86, 94, 103, },

table3 = { 0, 0, 0, 1, 3, 5, 9, 14, 19, 25, 32, 40, 55, 73, 91, 110, },

table4 = { 0, 0, 0, 0, 0, 1, 2, 4, 6, 8, 11, 14, 26, 51, 77, 102, },

[00160] As described above in paras [00150]-[00153] the method of table subsampling can drop N bits from table values to reduce table size. As also mentioned in these paras N may be different depending on QP used for certain table generation and selected THR value for this table. For example for lower QP values the filtering parameter σ is relatively lower than for higher QPs. Thus the absolute value of THR can be reduced without sacrificing of performance. Furthermore to keep table size same for all QPs in the set (that is beneficial for simplifying implementation) and reduce subsampling rate for lower QPs (which correspond for less compression level and better reconstructed picture quality) it may be beneficial to reduce the amount of dropped bits N in comparison to other QP tables, e.g. by setting N for lower QP equal to 2 and setting THR to 64. The FIG. 16 illustrates example of filter transfer functions for five QPs in the set based on corresponding tables (from 0 to 4) with N set to 2 for first table (corresponding to lower QP range) and N set to 3 for other tables. This example also includes method of using *AuxiliaryFunction* for table 3 and table 4 generation as described in paras [00159]-[00160]. For table 4 straight-line auxiliary function crosses the x-axis at the value of 11. For table 4 straight-line auxiliary function crosses the x-axis at the value of 9. The tables used in this example have values according to:

table0 = { 0, 0, 2, 6, 10, 14, 19, 23, 28, 32, 36, 41, 45, 49, 53, 57, },

table1 = { 0, 0, 5, 12, 20, 29, 38, 47, 56, 65, 73, 82, 90, 98, 107, 115, },

table2 = { 0, 0, 1, 4, 9, 16, 24, 32, 41, 50, 59, 68, 77, 86, 94, 103, },

table3 = { 0, 0, 0, 1, 3, 5, 9, 14, 19, 25, 32, 40, 55, 73, 91, 110, },

table4 = { 0, 0, 0, 0, 0, 1, 2, 4, 6, 8, 11, 14, 26, 51, 77, 102, },

[00161] As described above in paras [00150]-[00153] the method of table subsampling can drop N bits from table values to reduce table size. As also mentioned in these paras N may be different depending on QP used for certain table generation and selected THR value for this table. For example for higher QP values the filtering parameter σ is relatively higher than for lower QPs that may require on increase THR value to keep equation from para [0077] closer to 1. At the same time to keep LUT size same for all QPs in the set (which is beneficial due to simplification of implementation) and also considering that for higher QP values reconstructed picture has more distortions and increasing subsampling of LUT is acceptable due to subjectively unnoticeable effect of LUT subsampling in presence of strong compression artifacts the value N of least significant bits dropped may be increased to 4 e.g. for last and for second last table in the set. FIG. 17 illustrates example of filter transfer functions for five QPs in the set based on corresponding tables (from 0 to 4) with N set to 2 for first table (table 0 corresponding to lower QP range), N set to 4 for last and second last table (table 3 and table 4) and N set to 3 for other tables. In this example THR is set to 64 for first table generation, set to 256 for last and second last table and set to 128 for rest tables. This example also includes method of using *AuxiliaryFunction* for table 3 and table 4 generation as described in paras [00159]-[00160]. For table 3 straight-line auxiliary function crosses the x-axis at the value of 6. For table 4 straight-line auxiliary function crosses the x-axis at the value of 8. The tables used in this example have values according to:

table0 = { 0, 0, 2, 6, 10, 14, 19, 23, 28, 32, 36, 41, 45, 49, 53, 57, },
table1 = { 0, 0, 5, 12, 20, 29, 38, 47, 56, 65, 73, 82, 90, 98, 107, 115, },
table2 = { 0, 0, 1, 4, 9, 16, 24, 32, 41, 50, 59, 68, 77, 86, 94, 103, },
table3 = { 0, 0, 3, 9, 19, 32, 47, 64, 81, 99, 117, 135, 154, 179, 205, 230, },
table4 = { 0, 0, 0, 2, 6, 11, 18, 27, 38, 51, 64, 96, 128, 160, 192, 224, },

[00162] FIG. 18 is a block diagram of an apparatus 600 that can be used to implement various embodiments. The apparatus 600 may be the encoding apparatus as shown in FIG.1 and the decoding apparatus as shown in FIG. 2. Additionally, the apparatus 600 can host one or more of the described elements. In some embodiments, the apparatus 600 is equipped with one or more input/output devices, such as a speaker, microphone, mouse, touchscreen, keypad, keyboard, printer, display, and the like. The apparatus 600 may include one or more central processing

CLAIMS

1. A method for processing a block, wherein the block comprises a plurality of pixels, wherein the method comprises:
obtaining a quantization parameter (QP) of the block, wherein the block is a reconstructed block;
obtaining a threshold (THR) based on the QP; and
obtaining a look up table (LUT) based on QP, so as to generate a filtered block based on the threshold and the look up table.

2. The method of claim 1, wherein the method further comprises:
scanning a current pixel of the block and its neighboring pixels of the current pixel according to a predefined scan template;
obtaining spectrum components by performing transform for the current pixel and its neighboring pixels;
obtaining filtered spectrum components by using the look up table and the spectrum components.

3. The method of claim 2, a filtered spectrum component $F(i, \sigma)$ is derived by:

$$F(i, \sigma) = \begin{cases} R(i) & , \text{Abs}(R(i)) \geq THR \\ LUT(R(i), \sigma), & R(i) > 0 \\ -LUT(-R(i), \sigma), & R(i) \leq 0 \end{cases}$$

wherein (i) is an index of a spectrum component, $R(i)$ is the spectrum component corresponding to (i) index, σ is a filtering parameter, $LUT(R(i), \sigma)$ is an array in the look up table, and THR is the threshold being derived from the QP.

4. The method of claim 3, wherein $LUT(R(i), \sigma) = \frac{R(i)^3}{R(i)^2 + m * \sigma^2}$, m is normalization constant equal to number of spectrum components.

5. The method of claim 3 or 4, wherein the filtering parameter is derived based on the QP.

6. The method of claim 5, wherein the filtering parameter σ is derived by

$$\sigma = k * 2^{(n*(QP-s))}$$

, wherein QP is the codec quantization parameter, k , n and s are constants.

7. The method of claim 6, wherein $k = 2.64$, $n = 0.1296$ and $s = 11$.

8. The method of any one of claims 1-7, wherein the threshold is derived by:

$$\frac{THR^2}{THR^2 + m * \sigma^2} = C,$$

wherein C is a value close to 1, σ is a filtering parameter, and m is normalization constant equal to number of spectrum components.

9. The method of any one of claims 1-8, wherein the method further comprises:

choose a set of QPs to obtain the look up table, wherein the set of QPs include a first QP corresponding to (i) index and a second QP corresponding to (i+1) index, and wherein the first QP and the second QP have an interval more than 1.

10. The method of claim 9, wherein the interval is constant and equal to 8, 10 or 16.

11. The method of any one of claims 1-10, wherein the method further comprises:

dropping N bits from table values of the look up table, N is an integer.

12. The method of claim 11, wherein access to element of LUT is as follows

$tbl[(z + HTDF_TBL_RND) \gg HTDF_TBL_SH]$,

where HTDF_TBL_SH is equal to N, and $HTDF_TBL_RND = 1 \ll (HTDF_TBL_SH - 1)$, wherein z is an integer.

13. The method of claim 11, wherein access to element of LUT is as follows

$$LUT[(fHad[i] + (1 \ll (tblShift - 1))) \gg tblShift]$$

for positive values of fHad[i], and

$$-LUT[(-(fHad[i] + (1 \ll (tblShift - 1))) \gg tblShift)]$$

for negative values of fHad[i],

where tblShift is equal to N, and fHad[i] is spectrum components to be filtered.

14. The method of any one of claims 11-13, wherein N is 3.

15. The method of any one of claim 9, wherein for the first QP in a set of QPs, N is equal to 2, and wherein the set of QPs is used to obtain the look_up table; wherein for the last QP or for the last two QPs in the set of QPs, N is equal to 4; and wherein for the rest QPs from the set of QPs, N is equal to 3.

16. The method of any one of claims 11-13, N is defined as follows;

$$tblShift = tblThrLog2[qpIdx] - 4$$

$$tblThrLog2[5] = \{ 6, 7, 7, 8, 8 \}$$

where tblShift is equal to N, and qpIdx is derived based on QP.

17. The method of claim 16, qpIdx is derived based on QP as follows;

$$\text{if(pred_mode_flag[xCb][yCb] = 0 \&\& nCbW = nCbH \&\& min(nCbW, nCbH) \geq 32)}$$

$$qpIdx = \text{Clip3}(0, 4, (QpY - 28 + (1 << 2)) >> 3)$$

else

$$qpIdx = \text{Clip3}(0, 4, (QpY - 20 + (1 << 2)) >> 3)$$

where QpY is the current block QP, xCb, yCb defines the position of the current block on the picture (x, y), pred_mode_flag[xCb][yCb] defines prediction mode of the current block, if equal to 0 the prediction is inter, otherwise intra, nCbW, nCbH is the width and height of the current block correspondingly.

18. The method of any one of claims 11-13, wherein N is dependent on QP value.

19. The method of any one of claims 1-18, wherein the LUT is generated based on an auxiliary function of at least some quantization parameters (QPs).

20. The method of claim 19, wherein the LUT $LUT(R_i, \sigma)$ is derived by:

$$LUT(R_i, \sigma) = \max \left(\frac{R_i^3}{R_i^2 + m * \sigma^2}, AuxiliaryFunc_{\sigma}(i) \right),$$

wherein $AuxiliaryFunc_{\sigma}(i)$ represents the auxiliary function, and the auxiliary function has a value equal to THR at i corresponding to an argument of last LUT + 1.

21. The method of claim 20, wherein the auxiliary function is a straight line equation through points (i,THR) and (a, 0), where $a > 0$ and a depends on QP value.

22. The method of claim 21, wherein the LUT is defined as follows

$$\text{setOfLUT}[5][16] =$$

```
{ 0, 0, 2, 6, 10, 14, 19, 23, 28, 32, 36, 41, 45, 49, 53, 57, },
{ 0, 0, 5, 12, 20, 29, 38, 47, 56, 65, 73, 82, 90, 98, 107, 115, },
{ 0, 0, 1, 4, 9, 16, 24, 32, 41, 50, 59, 68, 77, 86, 94, 103, },
{ 0, 0, 3, 9, 19, 32, 47, 64, 81, 99, 117, 135, 154, 179, 205, 230, },
```

{ 0, 0, 0, 2, 6, 11, 18, 27, 38, 51, 64, 96, 128, 160, 192, 224, }.

23. The method of claim 21, wherein the LUT is defined as follows

```
setOfLUT[ 5 ][ 16 ] =  
{ 0, 0, 2, 6, 10, 14, 19, 23, 28, 32, 36, 41, 45, 49, 53, 57, },  
{ 0, 0, 5, 12, 20, 29, 38, 47, 56, 65, 73, 82, 90, 98, 107, 115, },  
{ 0, 0, 1, 4, 9, 16, 24, 32, 41, 50, 59, 68, 77, 86, 94, 103, },  
{ 0, 0, 0, 1, 3, 5, 9, 14, 19, 25, 32, 40, 55, 73, 91, 110, },  
{ 0, 0, 0, 0, 0, 1, 2, 4, 6, 8, 11, 14, 26, 51, 77, 102, }.
```

24. The method of claim 21, wherein the LUT is defined as follows

```
setOfLUT[ 5 ][ 16 ] =  
{ 0, 2, 10, 19, 28, 36, 45, 53, 61, 70, 78, 86, 94, 102, 110, 118, },  
{ 0, 0, 5, 12, 20, 29, 38, 47, 56, 65, 73, 82, 90, 98, 107, 115, },  
{ 0, 0, 1, 4, 9, 16, 24, 32, 41, 50, 59, 68, 77, 86, 94, 103, },  
{ 0, 0, 0, 1, 3, 5, 9, 14, 19, 25, 32, 40, 55, 73, 91, 110, },  
{ 0, 0, 0, 0, 0, 1, 2, 4, 6, 8, 11, 14, 26, 51, 77, 102, }.
```

25. The method of claim 2, wherein the transform is a Hadamard transform.

26. The method of claim 2, wherein the transform is a 1D transform.

27. A decoder comprising processing circuitry for carrying out the method according to any one of claims 1-26.

28. An encoder comprising processing circuitry for carrying out the method according to any one of claims 1-26.

29. A decoder, comprising:

a memory storage comprising instructions; and

one or more processors in communication with the memory, wherein the one or more processors execute the instructions to carry out the method according to any one of claims 1-26.

30. An encoder, comprising:

a memory storage comprising instructions; and
one or more processors in communication with the memory, wherein the one or more processors execute the instructions to carry out the method according to any one of claims 1-26.

31. A computer program product comprising program code for performing the method according to any one of claims 1-26 when executed on a computer or a processor.