executed by the processors, configures the encoder to carry out the method according to any one of the ninth aspect, the third aspect and any one of implementation of the ninth aspect.

The fourteenth aspect of the present invention provides a non-transitory computer-readable medium carrying a program code which, when executed by a computer device, causes the computer device to perform the method according to any one of the ninth aspect and any one of implementation of the ninth aspect.

The fifteenth aspect of the present invention provides a non-transitory storage medium comprising a bitstream encoded/decoded by the method of any one of the above embodiments.

BRIEF DESCRIPTION OF THE DRAWINGS

In the following embodiments of the invention are described in more detail with reference to the attached figures and drawings, in which:

Fig. 1A is a block diagram showing an example of a video coding system configured to implement embodiments of the invention;

Fig. 1B is a block diagram showing another example of a video coding system configured to implement embodiments of the invention;

Fig. 2 is a block diagram showing an example of a video encoder configured to implement embodiments of the invention;

Fig. 3 is a block diagram showing an example structure of a video decoder configured to implement embodiments of the invention;

Fig. 4 is a block diagram illustrating an example of an encoding apparatus or a decoding apparatus;

prepares, or provides the various coding operations. The inclusion of the coding module 470 therefore provides a substantial improvement to the functionality of the video coding device 400 and effects a transformation of the video coding device 400 to a different state. Alternatively, the coding module 470 is implemented as instructions stored in the memory 460 and executed by the processor 430.

The memory 460 may comprise one or more disks, tape drives, and solid-state drives and may be used as an over-flow data storage device, to store programs when such programs are selected for execution, and to store instructions and data that are read during program execution. The memory 460 may be, for example, volatile and/or non-volatile and may be a read-only memory (ROM), random access memory (RAM), ternary content-addressable memory (TCAM), and/or static random-access memory (SRAM).

Fig. 5 is a simplified block diagram of an apparatus 500 that may be used as either or both of the source device 12 and the destination device 14 from Fig. 1A according to an exemplary embodiment**.**

A processor 502 in the apparatus 500 can be a central processing unit. Alternatively, the processor 502 can be any other type of device, or multiple devices, capable of manipulating or processing information now-existing or hereafter developed. Although the disclosed implementations can be practiced with a single processor as shown, e.g., the processor 502, advantages in speed and efficiency can be achieved using more than one processor.

A memory 504 in the apparatus 500 can be a read only memory (ROM) device or a random access memory (RAM) device in an implementation. Any other suitable type of storage device can be used as the memory 504. The memory 504 can include code and data 506 that is accessed by the processor 502 using a bus 512. The memory 504 can further include an operating system 508 and application programs 510, the application programs 510 including at least one program that permits the processor 502 to perform the methods described here.

Alternatively, for the block classification of block beginning at line K, only the lines M, N, O, P, Q are used. Line L could be further used for the classification, in an example, calculates gradients for every set of 2 lines which makes it cumbersome to include line L in the classification.

The second solution is to use simple padding. The line K is padded to line J, I, H and then the lines beginning from H till Q can be used for the block classification.

Solution 2 (Symmetric truncated ALF block classification)

As shown in Fig. 10, an implementation refer to modified ALF classification to avoid ALF Luma Line Buffers (Symmetric variant) is disclosed, the symmetric truncated ALF block classification only uses the inner 4 samples (a,b,c,d) to classify the given block.

In an example,

Solution 1:

For block classification of 4 x 4 block immediate above VB (Fig. 10a):

Only use samples above VB in classification. Mainly Samples a, b, c, d are used.

For block classification of 4 x 4 block immediate below VB (Fig. 10b):

Only use samples below VB in classification. Mainly Samples a, b, c, d are used.

In Fig. 10a, for the block starting at line G, only the samples belonging to lines H and I (samples a, b, c, d) are used in the gradient computations. No additional padding is required in this case. Similarly in Fig. 10b, for the block starting at line K, only the samples belonging to lines L and M (samples a, b, c, d) are used in the gradient computations. No additional padding is required in this case. This solution has the advantage that the gradient is computed using a symmetric block shape which may be advantageous for subjective picture quality, especially when the picture content changes frequently inside a given local window. Also the solution is computationally less complex as it computes gradient using only 4 samples (a, b, c, d).

– If VirtualBoundariesDisabledFlag is equal to 1, and xCtb + x − VirtualBoundariesPosX[n] is greater than or equal to 0 and less than 3 for any n = 0..VirtualBoundariesNumVer − 1, the following applies:

clipLeftPos = VirtualBoundariesPosX[n] (1470)

– Otherwise, if x is less than 3, and one or more of the following conditions are true, the variable clipLeftPos is set equal to xCtb:
  – The left boundary of the current coding tree block is the left boundary of the tile, and loop_filter_across_tiles_enabled_flag is equal to 0.
  – The left boundary of the current coding tree block is the left boundary of the slice, and loop_filter_across_slices_enabled_flag is equal to 0.
  – The left boundary of the current coding tree block is the left boundary of the subpicture, and loop_filter_across_subpic_enabled_flag[SubPicIdx] is equal to 0.

The variable clipRightPos is modified as follows:

If VirtualBoundariesDisabledFlag is equal to 1, and VirtualBoundariesPosX[n]− xCtb − x is greater than 0 and less than 5 for any n = 0..VirtualBoundariesNumVer − 1, the following applies:

clipRightPos = VirtualBoundariesPosX[n] (1471)

Otherwise, if CtbSizeY − x is less than 5, and one or more of the following conditions are true, the variable clipRightPos is set equal to xCtb + CtbSizeY:

The right boundary of the current coding tree block is the right boundary of the tile, and loop_filter_across_tiles_enabled_flag is equal to 0.

The right boundary of the current coding tree block is the right boundary of the slice, and loop_filter_across_slices_enabled_flag is equal to 0.

The right boundary of the current coding tree block is the right boundary of the subpicture, and loop_filter_across_subpic_enabled_flag[SubPicIdx] is equal to 0.

The variable clipTopLeftFlag and clipBotRightFlag are modified as following:

If the coding tree block covering the luma position (xCtb, yCtb) and the coding tree block covering the luma position (xCtb − CtbSizeY, yCtb − CtbSizeY) belong to different slices, and loop_filter_across_slices_enabled_flag is equal to 0, clipTopLeftFlag is set equal to 1.

If the coding tree block covering the luma position (xCtb, yCtb) and the coding tree block covering the luma position (xCtb + CtbSizeY, yCtb + CtbSizeY) belong to

CLAIMS

1. A method of filtering a sample value of a picture,

wherein the method comprises:

obtaining a reconstructed sample value for a block of the picture;

obtaining filter coefficients for adaptive loop filtering according to a bitstream;

obtaining a sum value according to the filter coefficients and the reconstructed sample value for the block;

rounding the sum value according to a vertical position of a sample in the block, to obtain a rounded sum value;

obtaining a filter reconstructed sample value for the block according to the rounded sum value.

2. The method of claim 1, wherein the rounded sum value is equal to (sum + (1 << (alfShiftY − 1))) >> alfShiftY, wherein sum is the sum value, alfShiftY is a variable which is determined based on the vertical position of the sample.

3. The method of claim 2, wherein alfShiftY is equal to 10 when the vertical position of the sample is satisfied any one of the below conditions:

y = = CtbSizeY – t-1, or

y = = CtbSizeY – t, or

y = = ctbHeightC – t, or

y = = ctbHeightC – t-1,

wherein y is the vertical position of the sample, CtbSizeY is a coding tree block, CTB, size, ctbHeightC is the height of the CTB, t is an integer value.

4. The method of claim 2 or 3, wherein alfShiftY is equal to 7 when the vertical position of the sample is not satisfied any one of the below conditions:

y = = CtbSizeY – t-1, or

y = = CtbSizeY – t, or

y = = ctbHeightC – t, or

y = = ctbHeightC – t-1,

wherein y is the vertical position of the sample, CtbSizeY is a coding tree block, CTB, size, ctbHeightC is the height of the CTB, t is an integer value.

5. The method of any one of claims 2 to 4, wherein when y is equal to 0, alfShiftY is equal to 10.

6. The method of any one of claims 2 to 5, wherein when y is not equal to 0, alfShiftY is equal to 7.

7. The method of any one of claims 1 to 6, wherein the vertical position of the sample in the block is a chroma sample position or is a luma sample position.

8. The method of any one of claims 1 to 7, wherein the rounding process comprises:

adding a value which is determined based on the vertical position to the sum value in order to obtain an added sum value, and

shifting the added sum value based on the vertical position.

9. A video decoding apparatus, comprising:

a reconstructing module, which is configured to obtain a reconstructed sample value for a block of a picture;

a parsing module, which is configured to obtain filter coefficients for adaptive loop filtering according to a bitstream;

a calculating module, which is configured to obtain a sum value according to the filter coefficients and the reconstructed sample value for the block;

a rounding module, which is configured to round the sum value according to a vertical position of a sample in the block, to obtain a rounded sum value;

a filtering module, which is configured to obtain a filter reconstructed sample value for the block according to the rounded sum value.

10. The video decoding apparatus of claim 9, wherein the rounded sum value is equal to (sum + (1 << (alfShiftY − 1))) >> alfShiftY, wherein sum is the sum value, alfShiftY is a variable which is determined based on the vertical position of the sample.

11. The video decoding apparatus of claim 10, wherein alfShiftY is equal to 10 when the vertical position of the sample is satisfied any one of the below conditions:

y = = CtbSizeY – t-1, or

y = = CtbSizeY – t, or

y = = ctbHeightC – t, or

y = = ctbHeightC – t-1,

wherein y is the vertical position of the sample, CtbSizeY is a coding tree block, CTB, size, ctbHeightC is the height of the CTB, t is an integer value.

12. The video decoding apparatus of claim 10 or 11, wherein alfShiftY is equal to 7 when the vertical position of the sample is not satisfied any one of the below conditions:

y = = CtbSizeY – t-1, or

y = = CtbSizeY – t, or

y = = ctbHeightC – t, or

y = = ctbHeightC – t-1,

wherein y is the vertical position of the sample, CtbSizeY is a coding tree block, CTB, size, ctbHeightC is the height of the CTB, t is an integer value.

13. The video decoding apparatus of any one of claims 10 to 12, wherein when y is equal to 0, alfShiftY is equal to 10.

14. The video decoding apparatus of any one of claims 10 to 13, wherein when y is not equal to 0, alfShiftY is equal to 7.

15. The video decoding apparatus of any one of claims 9 to 14, wherein the vertical position of the sample in the block is a chroma sample position or is a luma sample position.

16. The video decoding apparatus of any one of claims 9 to 15, wherein the rounding module is configured to

add a value which is determined based on the vertical position to the sum value in order to obtain an added sum value, and shift the added sum value based on the vertical position, to obtain the rounded sum value.

17. A decoder, comprising:

one or more processors; and

a non-transitory computer-readable storage medium coupled to the processors and storing programming for execution by the processors, wherein the programming, when executed by the processors, configures the decoder to carry out the method according to any one of the claims 1 to 8.

18. A non-transitory computer-readable medium comprising an encoded bitstream for a video signal decoded by performing the method of any one of the claims 1 to 8.

5