*IEIE Transactions on Smart Processing and Computing, vol. 6, no. 2, April 2017*
https://doi.org/10.5573/IEIESPC.2017.6.2.093



# Complexity Reduction of an Adaptive Loop Filter Based on Local Homogeneity

**Xiang Li, Yongjo Ahn, and Donggyu Sim**

Dept. of Computer Engineering, Kwongwoon University / Seoul, Korea {starxiang1945, yongjoahn, dgsim}@kw.ac.kr

* Corresponding Author: Donggyu Sim

*Received February 8, 2017; Accepted February 23, 2017; Published April 30, 2017*

** Regular Paper*

***Abstract*:** This paper proposes an algorithm for adaptive loop filter (ALF) complexity reduction in the decoding process. In the original ALF algorithm, filtering for I frames is performed in the frame unit, and thus, all of the pixels in a frame are filtered if the current frame is an I frame. The proposed algorithm is designed on top of the local gradient calculation. On both the encoder side and the decoder side, homogeneous areas are checked and skipped in the filtering process, and the filter coefficient calculation is only performed in the inhomogeneous areas. The proposed algorithm is implemented in Joint Exploration Model (JEM) version 3.0 future video coding reference software. The proposed algorithm is applied for frame-level filtering and intra configuration. Compared with the JEM 3.0 anchor, the proposed algorithm has 0.31%, 0.76% and 0.73% bit rate loss for luma (Y) and chroma (U and V), respectively, with about an 8% decrease in decoding time.

***Keywords*:** High efficiency video coding (HEVC), Future video coding (FVC), Adaptive loop filter (ALF)

## 1. Introduction

The High Efficiency Video Coding (HEVC) standard is currently the newest video coding standard, which was finalized in January 2013 by the Joint Collaborative Team on Video Coding (JCT-VC). JCT-VC is the joint expert team formed by the International Telecommunication Union-Telecommunication (ITU-T) Video Coding Expert Group (VCEG) and the International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) Moving Picture Expert Group (MPEG) standardization organizations. HEVC was developed to meet the growing need for high-compression video applications, video conferences, digital storage media, television broadcasting, internet streaming, and information communications [1-3]. The HEVC coding layer employs the same hybrid coding approach as previous video coding standards since H.261, but it has realized significantly improved compression performance (over a 50% bit-rate reduction for equal perceptual video quality), compared to the existing H.264/AVC video coding standard [4].

However, with the increasing diversity of media services, the amount of streaming data that needs to be transferred through the internet is sharply rising. Because network resources are limited, more effective, faster, and more robust video compression algorithms are demanded. In October 2015, ITU-T VCEG (Q6/16) and ISO/IEC MPEG joined together to study the need for standardization of future video coding technology to obtain a significantly high compression capability, compared with the current HEVC standard. The goal of future video coding is mainly focused on high resolution (over 8K) video coding, supporting high bit–depth (12-bit) video coding, high frame rates (over 120 frames per second), a wide color gamut, and a high dynamic range. At the current stage of standardization, 10-bit coding is already supported. Some of the tools added in Joint Exploration Model (JEM) software [5] are more efficient when coding higher resolution sequences.

In the early stages of HEVC standardization, the adaptive loop filter (ALF) was once applied in the HM reference software [6] before version 7.0, and the ALF conducted the filter process pixel by pixel by employing a Wiener filter, which is also referred to as a minimum mean square error filter. The ALF was designed to minimize the error between the original frame and the reconstructed frame, which requires huge amounts of computation to obtain the filter coefficients. For this reason, the ALF was finally removed after HM 8.0. However, even if the ALF

filter coefficient calculation process is of high computational complexity, it achieves good performance; therefore, it was proposed again at the future video coding standardization meeting.

In the original ALF algorithm, filtering for I frames is performed in the frame unit, and thus, all of the pixels in a frame are filtered if the current frame is an I frame. The proposed algorithm is designed on top of the local gradient calculation. On both the encoder side and the decoder side, the homogeneous areas are checked and skipped in the filtering process, and filter coefficient calculation is only performed in the inhomogeneous areas. The proposed algorithm is implemented in JEM version 3.0. The proposed algorithm is applied for frame-level filtering and intra configuration, and compared with the JEM 3.0 anchor, the proposed algorithm has a 0.31%, 0.76%, and 0.73% bit-rate loss for luma (Y) and chroma (U and V), respectively, with about an 8% decrease in decoding time.

In this paper, Section 2 covers some previous work on the ALF. Section 3 gives an overall description of the existing ALF filter algorithm, and details about the proposed ALF algorithm method. Section 4 shows the experiment performance and analysis about the performance. Finally, Section 5 gives the conclusion to this paper.

## 2. JEM and Adaptive Loop Filters

During the exploration process for future video coding, the corresponding JEM reference software was released. Until now, JEM reference software has released three main versions; the experiment in this paper is based on the newest version, JEM 3.0. The reference software consists of an encoder and a decoder, where the encoder encodes the input frame into a bitstream, the bitstream is transmitted to the decoder, and then the decoder reconstructs the frame depending on the video coding specifications. Since artifacts can be generated in the signal, such as blocking artifacts and ringing artifacts during the encoding process, to remove those artifacts as well as to provide a better reference for the following encoding frames, the in-loop filtering process is placed before the reconstructed frame is sent to the decoded picture buffer (DPB).

### 2.1 JEM

In the current stage of future video coding exploration, the main coding structure remains unchanged; only some tools have been added to either enhance coding efficiency or reduce the complexity of the existing HEVC reference software. The new coding tools and features are listed in Table 1 [7].

In the encoder, the input frame is fed into the encoder, and then the frame is split into smaller square-shaped blocks; in HEVC, the split is conducted in quad-tree style, and in future video coding, a quad-tree plus binary tree is applied in block structure splitting; thus, a square block can not only be split into four smaller blocks, but can also be split only horizontally or only vertically. The splitting

**Table 1. Tools Added into JEM Reference Software.**

| Module | Tools |
|---|---|
| Block structure | Quadtree plus binary tree (QTBT) block structure |
| Intra prediction module | 65 intra prediction directions |
| | 4-tap interpolation filter for intra prediction |
| | Boundary filter applied to other directions, in addition to horizontal and vertical |
| | Cross-component linear model (CCLM) prediction |
| | Position-dependent intra prediction combination (PDPC) |
| | Adaptive reference sample smoothing |
| Inter prediction module | Sub-PU level motion vector prediction |
| | Locally adaptive motion vector prediction |
| | 1/16 pel-motion vector storage accuracy |
| | Overlapped block motion compensation (OBMC) |
| | Local illumination compensation (LIC) |
| | Affine motion prediction |
| | Pattern-matched motion vector derivation |
| | Bi-directional optical flow (BIO) |
| Trans-formation module | Explicit multiple core transform |
| | Mode-dependent non-separable secondary transforms |
| | Signal-dependent transforms (SDT) |
| In-loop filter | Adaptive loop filter |
| Entropy module | Context model selection for transform coefficient levels |
| | Multi-hypothesis probability estimation |
| | Initialization for context models |

information will be sent to the decoder. Some tools were also merged into the prediction module for intra prediction, which generates the reference to the current coding frame. The number of prediction modes has been increased to 65 to suit high-resolution video sequences, and a cross-component linear model raises the idea that the luma prediction mode can be used to derive the chroma prediction mode. For inter prediction, which generates the reference blocks from other frames, the motion vector fetching method is enhanced, like Advanced Temporal Motion Vector Prediction (ATMVP) and Spatial-Temporal Motion Vector Prediction (STMVP). Also, some techniques that were not used in standard video coding have been added, like Local Illumination Compensation (LIC), affine mode, and Bi-directional Optical Flow (BIO), so that, with these tools added, inter prediction becomes more accurate than before. After prediction, the residual will be scaled and quantized, the transform style also becomes more variable, several transform cores are added in the “explicit multiple core transform” tool, and the Karhunen-Loeve Transform (KLT) technique is added in the SDT tool. Then, entropy-coded and transmitting the bits to the decoder side, in the entropy coding, the existing Context Adaptive Binary Arithmetic Coding (CABAC) has been enhanced. In the filtering, the biggest change is that an adaptive loop filter has been added after the sample

adaptive offset (SAO) process. The in-loop filter module consists of a de-blocking filter and the SAO filter. The de-blocking filter removes the blocking artifacts generated on the boundary of the blocks. Those artifacts are mainly caused by quantization, since different blocks have different quantization levels, which lead to the discontinuity in the block boundaries. The SAO filter mainly focuses on subjective quality enhancement and ringing artifacts removal by applying the offset to the block based on the feature of the blocks.

## 2.2 ALF

Plenty of work has been done to reduce the complexity of the ALF. Some of it exploits the similarity between the neighboring frames. Zhang et al. [8] proposed the concept of a filter coefficient bank that saves the ALF filter coefficients that come from the previous encoded frame. The current frame can reuse the filter coefficient saved in the filter coefficient bank to save both encoding time and decoding time. There are two filter coefficient banks established to save the previous two frames' filter coefficients. One flag indicates which filter coefficient will be chosen, and it is decided according to the rate distortion performance. In another paper [9], Zhang et al. pre-judge the filter design mode for every largest coding unit (LCU), based on the spatial correlation and texture information of video content, and the time complexity of the adaptive loop filter can be significantly reduced by skipping some unnecessary modes. Budagavi et al. [10] proposed two ALF filter sets, N×7 and N×5, which can reduce the vertical size of the filter shape, and thereby, the line buffer size and memory size can be reduced. A one-pass encoding algorithm was proposed [11] that can estimate filtering distortion without performing a real filtering process, and the encoding pass can be effectively reduced from 16 to 1, which is significantly efficient for hardware design. Park et al. [12] proposed an adaptive selection method where the ALF can be applied more efficiently in the frame level to reduce decoding time. This algorithm takes advantage of the fact that the ALF is more effective when applied to a frame with high rate distortion cost values.

# 3. Proposed Complexity Reduction of the ALF Based on Local Homogeneity

The ALF minimizes the mean square error between original samples and decoded samples by using a Wiener-based adaptive filter, and the ALF is located in the last stage of each picture, before the reconstructed picture is sent to the DPB. The encoder structure with the ALF is shown in Fig. 1. The reconstructed picture after being filtered by the SAO is fed into the ALF, and the ALF is the final process to fix the artifacts from the previous stages. The whole process of the ALF consists of Wiener filter coefficient derivation, a filter shape decision, the filtering process, and the coefficient coding process [13].

The Wiener filter, which was formulated by Norbert Wiener, set the foundation for data-dependent linear least

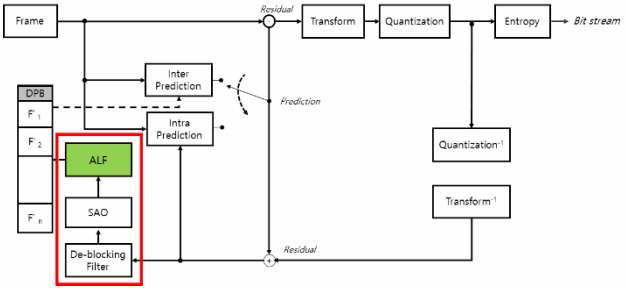



**Fig. 1. HEVC Encoding Structure with ALF Applied.**

square error filters. A Wiener filter can be a finite-duration impulse filter (FIR), and it can also be an infinite-duration impulse filter (IIR), since the filter coefficients for image processing are limited [14]:

$$e(m) = x(m) - \tilde{x}(m) \tag{1}$$

where m indicates the position of the pixel, x(m) represents the original signal, and $\tilde{x}$(m) represents the reconstructed signal. Then e(m) is the error between the original signal and the reconstructed signal, and the goal of the Wiener filter is to minimize this error to make the reconstructed signal as close to the original signal as possible.

According to the Wiener filter derivation [15], the Wiener filter can finally be written in the form of a matrix, as shown below.

$$\begin{bmatrix} \sum_{[r]} t[r+p_0][r+p_0] & \sum_{[r]} t[r+p_1][r+p_0] & \cdots & \sum_{[r]} t[r+p_{N-1}][r+p_0] \\ \sum_{[r]} t[r+p_0][r+p_1] & \sum_{[r]} t[r+p_1][r+p_1] & \cdots & \sum_{[r]} t[r+p_{N-1}][r+p_1] \\ \cdots & \cdots & \cdots & \cdots \\ \sum_{[r]} t[r+p_0][r+p_{N-1}] & \sum_{[r]} t[r+p_1][r+p_{N-1}] & \cdots & \sum_{[r]} t[r+p_{N-1}][r+p_{N-1}] \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \cdots \\ c_{N-1} \end{bmatrix} = \begin{bmatrix} \sum_{[r]} s[r][r+p_0] \\ \sum_{[r]} s[r][r+p_1] \\ \cdots \\ \sum_{[r]} s[r][r+p_{N-1}] \end{bmatrix}$$

The matrix on the left represents autocorrelation of the reconstructed signal, and the one-dimensional matrix on the right represents the cross-correlation of the reconstructed signal and the original signal [16].

As the method illustrates above, the JEM reference software derives the filter coefficients by calculating the auto-correlation and cross-correlation of the reconstructed signal and the original signal before filtering the frame [17].

According to the different features of the pixels in a frame, a total of 25 coefficient sets will be derived for filtering a frame, and different coefficient sets are decided by the "direction" and "activity" of each pixel, where different filter coefficients will be applied to a different "direction" and "activity" [7].

$$C = 5D + A \tag{2}$$

In (2), where C is the classification index, D is the "direction", and A is the quantized "activity", the calculation of "direction" and "activity" is based on the following method.

First, the vertical gradient of each pixel is calculated:

$$V_{k,l} = \left|2R(k,l) - R(k,l-1) - R(k,l+1)\right| \quad (3)$$

Secondly, the horizontal gradient of each pixel is calculated:

$$H_{k,l} = \left|2R(k,l) - R(k-1,l) - R(k+1,l)\right| \quad (4)$$

Third, the left diagonal gradient of each pixel is calculated as

$$D_{1_{k,l}} = \left|2R(k,l) - R(k-1,l-1) - R(k+1,l+1)\right| \quad (5)$$

Then, the right diagonal gradient of each pixel is

$$D_{2_{k,l}} = \left|2R(k,l) - R(k-1,l+1) - R(k+1,l-1)\right| \quad (6)$$

Next, according to the gradient of each direction, the vertical gradient of each 2×2 block can be derived as

$$g_v = \sum_{k=i-2}^{i+3}\sum_{l=i-2}^{j+3} V_{k,l} \quad (7)$$

The horizontal gradient of each 2×2 block is

$$g_h = \sum_{k=i-2}^{i+3}\sum_{l=i-2}^{j+3} H_{k,l} \quad (8)$$

Then, the left gradient of each 2 × 2 block is

$$g_{d1} = \sum_{k=i-2}^{i+3}\sum_{l=i-2}^{j+3} D_{1_{k,l}} \quad (9)$$

The right gradient of each 2 × 2 block is

$$g_{d2} = \sum_{k=i-2}^{i+3}\sum_{l=i-2}^{j+3} D_{2_{k,l}} \quad (10)$$

After the gradient for each area is calculated, the "direction" of each block is calculated, and each block will have a direction within the range 0 to 4 [34]. According to the process illustrated above, the "direction" can have a value in a range from 0 to 4. Then the "activity" can be calculated as

$$A = \sum_{k=i-2}^{i+3}\sum_{l=j-2}^{j+3} \left(V_{k,l} + H_{k,l}\right) \quad (11)$$

and A will be quantized to an integer ranging from 0 to 4. The final classification index can be decided depending on Eq. (11).

The filter shape decides the number of coefficients that will influence the filtering process. The filter shapes used in the JEM reference software are 5 × 5, 7 × 7, and 9 × 9 diamond shapes. (However, the shapes were 5 × 5, 7 × 7, and truncated 9 × 9 diamond in the previous JEM reference software.) In the implementation of ALF in the JEM reference software, the encoder chooses the best filter shape to be applied for the frame and signals to the decoder, and different frames can have different filter shapes.

The ALF filtering process can not only be applied in the frame level, but can also be applied to just some specified coding trees (CUs). In JEM reference software, the CU-level control filtering is applied for the P frames and B frames. CU-level controlling is the process that selects the CUs to be filtered, and turns off the filtering process for the remaining CUs. However, prior to the CU-controlling process, all of the pixels in a frame will be filtered once, and then, the sum of the squared differences (SSD) of the original CU and the CU before filtering (denoted as the reconstructed CU) will first be calculated:

$$SSD_1 = \sum_{k=0}^{N} \left(R_{ori} - R_{rec}\right)^2 \quad (12)$$

Then, the SSD of the original CU and the CU after filtering will be calculated:

$$SSD_2 = \sum_{k=0}^{N} \left(R_{ori} - R_{fil}\right)^2 \quad (13)$$

These two SSDs represent the quality of the CU before filtering and the quality after filtering, if

$$SSD_1 > SSD_2 \quad (14)$$

which indicates that the quality of the CU becomes better after filtering. Then, the current CU will be on for filtering; otherwise if

$$SSD_1 \le SSD_2 \quad (15)$$

it means the quality of the CU does not become better after filtering. In this case, the current CU will be off for filtering. By applied this process, an array consisting of "1" (for filtering on) and "0" can be constructed and sent to the decoder.

As shown in Fig. 2, before filtering, the direction and activity will be calculated for each pixel. Then, 25 filter coefficient sets will be generated according to the auto-correlation and cross-correlation of the pixels in a frame. After that, the filter process will be applied to luma components, and if the RD cost of a filtered frame is better than the reconstructed frame before filtering, the ALF for luma will be enabled; otherwise, it will be disabled. Only if the luma ALF filtering is enabled will the chroma filtering process be conducted. Finally, the filter coefficient information will be sent to the decoder. If chroma ALF filtering is processed, then, the finally result of filtering is compared with the reconstructed frame before filtering, and if the RD cost is smaller, which means the quality of the frame is better, then ALF luma filtering for the current frame will be set to on; otherwise, ALF luma filtering for the current frame will be turned off.

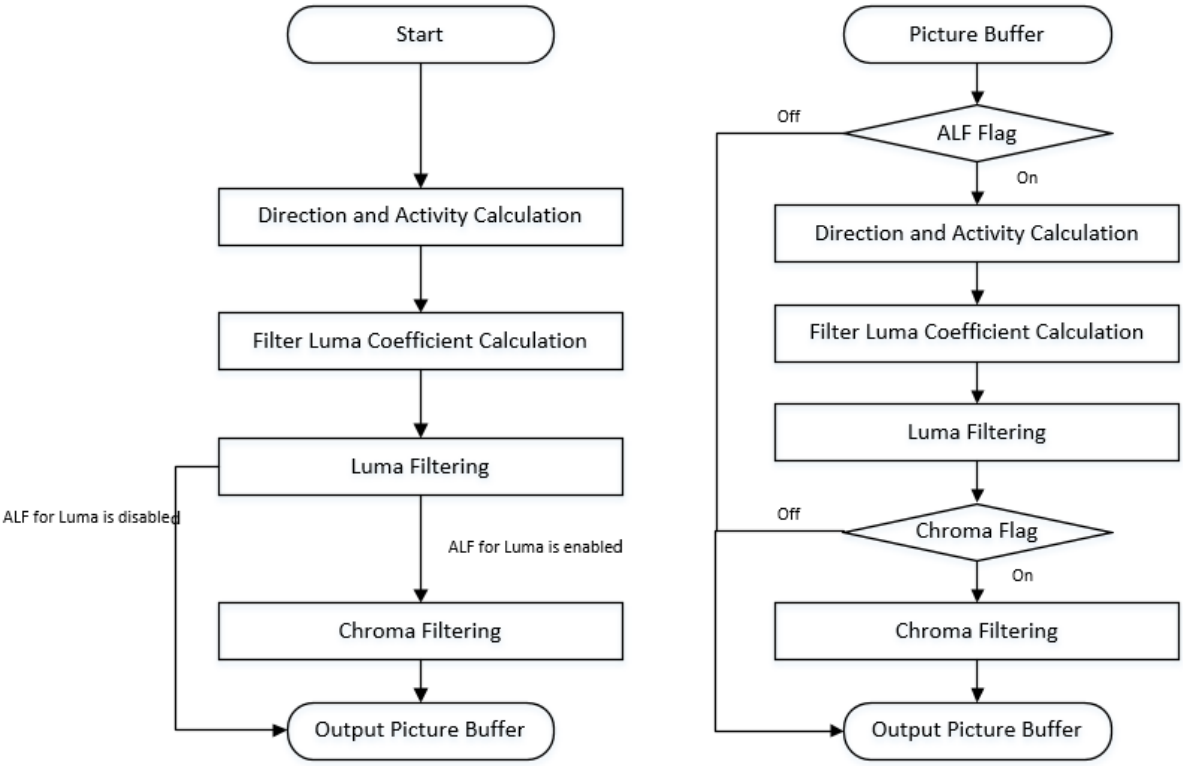



**Fig. 2. Existing ALF Processing Structure.**

In the decoder, the reconstructed frame is fed into the ALF process, but first of all, the ALF flag indicates whether the ALF is applied for the current frame or not, and if the ALF flag is off, then the ALF will be disabled, and if the ALF is enabled, the pixels will be filtered by using the filter coefficients sent from the encoder. Then, the same filtering result will be obtained as in the encoder. Otherwise, the ALF process will not be continued and skipped for the next stage.

In the ALF filtering process, for an I frame, all of the pixels within the frame will be filtered, comparing the frame after filtering with the frame before filtering. Some pixel values change, and some do not change. According to the analysis of those changed pixels and unchanged pixels, the unchanged pixels are mainly focused on smooth areas without edges or boundaries. However, the changed pixels are mainly concentrated in those unsmooth areas. Therefore, if those unchanged pixels are removed from the filtering process, the quality of the filtered frame will not be influenced, but some filtering time can be saved.

This paper proposes an algorithm to remove most of the smooth areas from the filtering process in order to save some filtering time. Therefore, a smooth area–checking process is applied before the filtering process, and the sum of the gradients of the horizontal direction and vertical direction is used to decide whether this area is smooth or not. The current 2 × 2 is determined as a smooth area if the sum of the horizontal gradient and the vertical gradient does not exceed a predefined threshold. The implementation flowchart is shown in Fig. 3.

In Fig. 3, the step blocks marked in gray are the steps that are different from the conventional ALF process. In the homogeneous area checking step, every 2 × 2 block will be checked, and if it is smooth, it is recorded. Then, in the filter coefficient calculation, only the inhomogeneous 2 × 2 blocks are involved in the calculation, and then only the inhomogeneous 2 × 2 blocks will be filtered. Also, the proposed algorithm is applied to chroma filtering. In the decoder, the homogeneous area will be checked before the filtering process, and then only the inhomogeneous areas will be filtered.

The method of how to determine the homogeneous areas will influence the performance of the proposed algorithm. In this paper, when the gradient of the area (2 × 2 block) exceeds predefined threshold Eq. (16), this area is defined as a homogeneous area; Otherwise, the area will be judged as an inhomogeneous area.

$$g_v + g_h + g_{d1} + g_{d2} < T \tag{16}$$

In Eq. (16), g_v, g_h, g_d1, and g_d2 represent the vertical gradient, horizontal gradient, left diagonal gradient, and left diagonal gradient of each block, respectively, and T is the predefined threshold. In this paper, the threshold is predefined as 300 by experience.

## 4. Performance Evaluation and Analysis

The proposed complexity reduction algorithm was implemented in JEM reference software version 3.0, and all the test sequences were selected from the Common Test Condition (CTC) [18], with other environment configurations listed in Table 2.

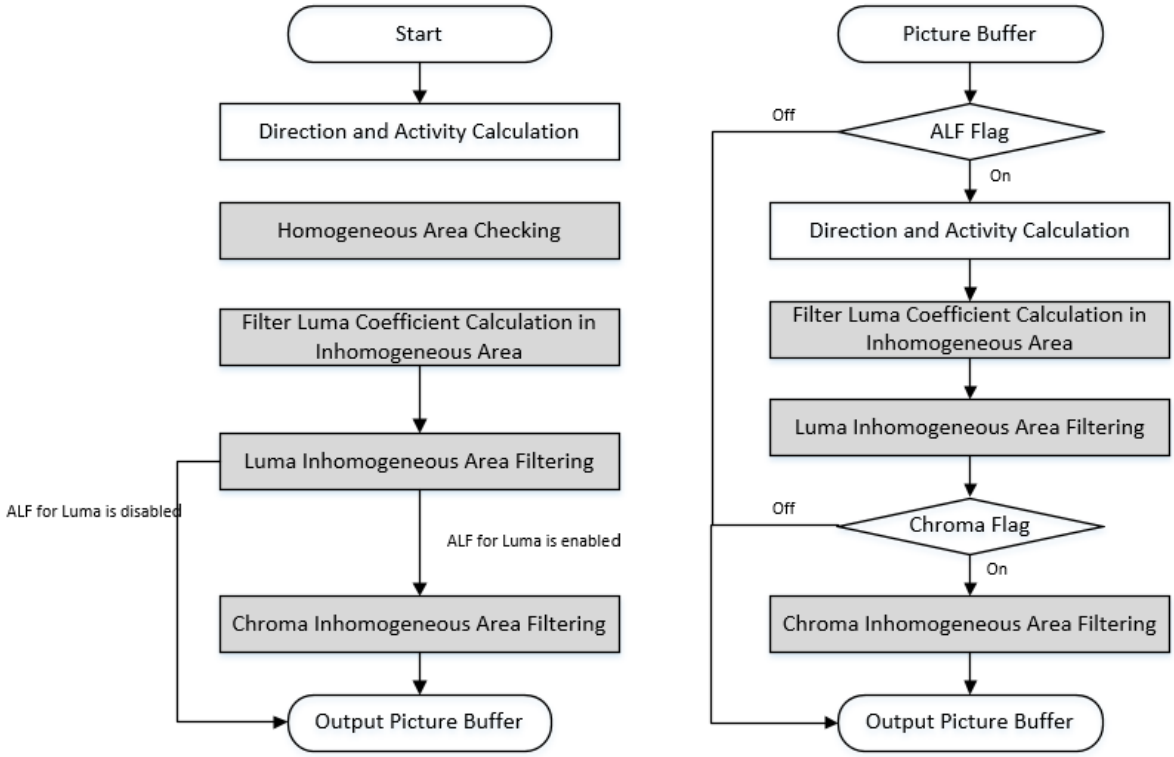



**Fig. 3. Proposed ALF Algorithm.**

**Table 2. Experimental Environment and Configuration Conditions.**

| Configuration | Content |
|---|---|
| CPU | Intel Core i7 |
| RAM | 16 GB |
| Operation System | Windows 10 |
| Coding Configuration | All Intra |
| Quantization Parameter | 22, 27, 32, 37 |
| Test Sequence Number | 21 |

**Table 3. Summary of Experimental Performance.**

| | Y | U | V | Average |
|---|---|---|---|---|
| Class A1 | 0.42 | 0.81 | 0.77 | 0.9 |
| Class A2 | 0.68 | 1.16 | 1.19 | 0.89 |
| Class B | 0.28 | 0.81 | 0.53 | 0.93 |
| Class C | 0.11 | 0.70 | 0.78 | 0.95 |
| Class D | 0.04 | 0.34 | 0.39 | 0.95 |
| Average | 0.31 | 0.76 | 0.73 | 0.92 |

The experimental performance summary is shown in Table 3. From the experimental performance, class A1 and class A2 have over a 90% decoding time decrease, since the sequences in classes A1 and A2 contain a lot of smooth areas; however, it should be noted that the coding efficiency performance was over 0.5% luma BD-rate loss and about 0.9% chroma BD-rate loss. And classes B, C, and D have about a 94% decoding time decrease because of the smaller resolution, and the smooth areas are not as numerous as in class A1 and class A2. The luma BD-rate loss and chroma BD-rate loss are just 0.14% and 0.6%, respectively, for U and Y components [19].

The detailed performance of each sequence is listed in Table 4. In Table 4, the experimental performance for each sequence of quantization parameters (QPs) 22, 27, 32, and 37 are listed. The "Reference psnr(db)" on the left is the anchor software: JEM reference software version 3.0 without any changes. "Tested psnr (db)" on the right is the tested software: JEM reference software with the proposed algorithm applied. There is a trend where, as the resolution increases, the decoding time reduction increases. This means that the higher the sequence resolution, the more the proposed algorithm can reduce decoding time. The same sequence coded by a different QP will have different reconstructed frames, and the higher the QP, the more the decoding time can be reduced, since the higher-QP reconstructed sequences have more smooth areas than the sequences coded by a lower QP.

## 5. Conclusion

This paper presents an algorithm to reduce the complexity of the adaptive loop filter in the decoder of JEM reference software version 3.0. The proposed algorithm checks the gradient of each block and judges it as a homogeneous block or not, and if a block is set as a homogeneous block, then this block will not be involved in the filter coefficient calculation but skipped in the filter process. The experiment was conducted with the sequences defined in the JEM software common test condition,

**Table 4. PSNR and Decoding Time Ratio of the Proposed Algorithm.**

| Sequences | | QP | Reference PSNR(db) | | | Tested PSNR(db) | | | Decoding Time |
|---|---|---|---|---|---|---|---|---|---|
| | | | Y | U | V | Y | U | V | |
| A1 | Tango | 22 | 41.82 | 49.54 | 48.25 | 41.82 | 49.51 | 48.24 | 0.93 |
| | | 27 | 40.29 | 48.51 | 46.61 | 40.28 | 48.51 | 46.58 | 0.85 |
| | | 32 | 39.50 | 47.47 | 45.15 | 39.47 | 47.44 | 45.13 | 0.84 |
| | | 37 | 38.39 | 46.55 | 43.86 | 38.34 | 46.41 | 43.83 | 0.83 |
| | Drum | 22 | 46.09 | 45.04 | 43.50 | 46.07 | 45.04 | 43.50 | 0.96 |
| | | 27 | 41.99 | 43.08 | 41.42 | 41.98 | 43.07 | 41.42 | 0.93 |
| | | 32 | 39.15 | 41.57 | 39.95 | 39.14 | 41.55 | 39.95 | 0.90 |
| | | 37 | 36.58 | 40.43 | 38.95 | 36.56 | 40.41 | 38.95 | 0.87 |
| | CampfireParty | 22 | 42.80 | 45.60 | 44.62 | 42.80 | 45.60 | 44.62 | 0.96 |
| | | 27 | 39.11 | 42.14 | 42.56 | 39.11 | 42.13 | 42.53 | 0.91 |
| | | 32 | 37.44 | 39.57 | 41.04 | 37.44 | 39.56 | 41.00 | 0.87 |
| | | 37 | 36.05 | 37.67 | 39.60 | 36.04 | 37.66 | 39.57 | 0.76 |
| | Toddle Fountain | 22 | 43.96 | 47.79 | 44.19 | 43.90 | 47.79 | 44.18 | 0.98 |
| | | 27 | 40.63 | 45.60 | 41.58 | 40.63 | 45.60 | 41.58 | 0.99 |
| | | 32 | 37.47 | 44.56 | 40.25 | 37.47 | 44.54 | 40.25 | 0.98 |
| | | 37 | 34.41 | 43.87 | 39.56 | 34.40 | 43.85 | 39.55 | 0.95 |
| A2 | CatRobot | 22 | 42.53 | 42.55 | 44.17 | 42.53 | 42.54 | 44.15 | 0.95 |
| | | 27 | 39.98 | 41.48 | 42.51 | 39.97 | 41.46 | 42.47 | 0.90 |
| | | 32 | 38.65 | 40.66 | 41.05 | 38.63 | 40.63 | 41.01 | 0.87 |
| | | 37 | 36.82 | 39.91 | 39.79 | 36.08 | 39.88 | 39.75 | 0.85 |
| | TrafficFlow | 22 | 40.93 | 47.43 | 47.04 | 40.93 | 47.41 | 47.03 | 0.95 |
| | | 27 | 39.32 | 45.60 | 45.47 | 39.31 | 45.59 | 45.44 | 0.90 |
| | | 32 | 37.99 | 44.15 | 44.03 | 37.98 | 44.14 | 44.01 | 0.88 |
| | | 37 | 36.39 | 43.12 | 42.90 | 36.37 | 43.10 | 42.88 | 0.85 |
| | DaylightRoad | 22 | 41.39 | 46.60 | 44.72 | 41.39 | 46.59 | 44.71 | 1.00 |
| | | 27 | 37.89 | 45.14 | 42.97 | 37.89 | 45.12 | 42.95 | 0.92 |
| | | 32 | 36.88 | 43.83 | 41.47 | 36.87 | 43.81 | 41.45 | 0.90 |
| | | 37 | 35.48 | 42.80 | 40.38 | 35.47 | 42.78 | 40.37 | 0.89 |
| | Rollercoaster | 22 | 47.32 | 48.85 | 47.50 | 47.27 | 48.82 | 47.50 | 0.89 |
| | | 27 | 45.24 | 46.61 | 45.32 | 45.16 | 46.59 | 45.30 | 0.85 |
| | | 32 | 43.01 | 44.78 | 43.94 | 42.93 | 44.74 | 43.90 | 0.84 |
| | | 37 | 40.47 | 43.27 | 42.89 | 40.39 | 43.19 | 42.84 | 0.83 |
| B | Kimono | 22 | 43.05 | 44.53 | 46.00 | 43.03 | 44.51 | 45.98 | 0.92 |
| | | 27 | 41.46 | 42.83 | 43.67 | 41.42 | 42.80 | 43.65 | 0.91 |
| | | 32 | 39.46 | 41.48 | 42.13 | 39.42 | 41.46 | 42.12 | 0.88 |
| | | 37 | 37.03 | 40.51 | 41.23 | 36.99 | 40.49 | 41.22 | 0.88 |
| | Parkcene | 22 | 42.30 | 43.45 | 44.02 | 42.30 | 43.39 | 44.00 | 0.98 |
| | | 27 | 39.27 | 41.13 | 41.28 | 39.27 | 41.07 | 41.27 | 0.97 |
| | | 32 | 36.23 | 39.31 | 39.73 | 36.23 | 39.27 | 39.27 | 0.93 |
| | | 37 | 33.35 | 38.07 | 38.97 | 33.35 | 38.04 | 38.97 | 0.93 |
| | Cactus | 22 | 41.31 | 41.53 | 44.35 | 41.31 | 41.53 | 44.35 | 1.00 |
| | | 27 | 38.40 | 39.60 | 42.37 | 38.40 | 39.59 | 42.35 | 0.95 |
| | | 32 | 36.27 | 38.60 | 40.73 | 36.26 | 38.59 | 40.70 | 0.94 |
| | | 37 | 33.93 | 37.79 | 39.47 | 33.92 | 37.78 | 39.43 | 0.90 |
| | BasketballDrive | 22 | 41.64 | 44.65 | 46.49 | 41.64 | 44.65 | 46.48 | 0.99 |
| | | 27 | 39.15 | 43.30 | 44.43 | 39.14 | 43.29 | 44.41 | 0.94 |
| | | 32 | 37.46 | 41.99 | 42.54 | 37.45 | 41.97 | 42.51 | 0.91 |
| | | 37 | 35.63 | 40.92 | 41.04 | 35.62 | 40.89 | 41.00 | 0.89 |
| | BQTerrace | 22 | 42.71 | 42.77 | 44.76 | 42.71 | 42.77 | 44.76 | 0.95 |
| | | 27 | 37.57 | 41.01 | 43.42 | 37.57 | 41.01 | 43.42 | 0.93 |
| | | 32 | 35.03 | 39.59 | 42.30 | 35.03 | 39.59 | 42.30 | 0.94 |
| | | 37 | 32.71 | 38.63 | 41.52 | 32.70 | 38.62 | 41.51 | 0.90 |

**Table 4. (Continued)**

| Sequences | | QP | Reference PSNR(db) | | | Tested PSNR(db) | | | Decoding Time |
|---|---|---|---|---|---|---|---|---|---|
| | | | Y | U | V | Y | U | V | |
| C | BasketballDrill | 22 | 42.63 | 44.21 | 45.04 | 42.63 | 44.20 | 45.03 | 0.97 |
| | | 27 | 39.27 | 41.84 | 42.44 | 39.26 | 41.83 | 42.43 | 0.95 |
| | | 32 | 36.27 | 39.95 | 40.35 | 36.26 | 39.93 | 40.33 | 0.95 |
| | | 37 | 33.63 | 38.45 | 38.64 | 33.61 | 38.43 | 38.61 | 0.92 |
| | BQMall | 22 | 42.51 | 44.46 | 46.18 | 42.51 | 44.45 | 46.16 | 0.96 |
| | | 27 | 39.77 | 42.30 | 43.74 | 39.77 | 42.28 | 43.72 | 0.95 |
| | | 32 | 36.79 | 40.43 | 41.64 | 36.78 | 40.41 | 41.62 | 0.94 |
| | | 37 | 33.73 | 39.01 | 40.03 | 33.72 | 38.99 | 40.01 | 0.94 |
| | PartyScene | 22 | 41.85 | 42.83 | 43.95 | 41.85 | 42.83 | 43.95 | 0.97 |
| | | 27 | 37.60 | 39.97 | 40.92 | 37.60 | 39.96 | 40.91 | 0.97 |
| | | 32 | 33.82 | 37.92 | 38.66 | 33.82 | 37.90 | 38.64 | 0.96 |
| | | 37 | 30.13 | 36.43 | 37.04 | 30.13 | 36.41 | 37.02 | 0.97 |
| | RaceHorse | 22 | 43.01 | 43.52 | 44.47 | 43.00 | 43.49 | 44.44 | 0.96 |
| | | 27 | 39.65 | 40.37 | 41.97 | 39.64 | 40.31 | 41.92 | 0.96 |
| | | 32 | 36.28 | 38.14 | 40.18 | 36.27 | 38.08 | 40.11 | 0.94 |
| | | 37 | 32.85 | 36.84 | 38.85 | 32.84 | 36.77 | 38.78 | 0.94 |
| D | BaskterballPass | 22 | 43.83 | 45.78 | 45.53 | 43.83 | 45.78 | 45.53 | 0.96 |
| | | 27 | 40.30 | 42.95 | 42.46 | 40.29 | 42.94 | 42.45 | 0.99 |
| | | 32 | 36.87 | 40.54 | 39.89 | 36.87 | 40.53 | 39.87 | 0.98 |
| | | 37 | 33.63 | 38.78 | 37.98 | 33.63 | 38.76 | 37.93 | 0.96 |
| | BQSquare | 22 | 42.02 | 43.63 | 44.84 | 42.01 | 43.63 | 44.84 | 0.92 |
| | | 27 | 37.61 | 40.90 | 42.25 | 37.61 | 40.90 | 42.24 | 0.94 |
| | | 32 | 33.76 | 39.28 | 40.49 | 33.76 | 39.27 | 40.48 | 0.98 |
| | | 37 | 30.25 | 38.30 | 39.18 | 30.25 | 38.30 | 39.17 | 0.97 |
| | BlowingBubbles | 22 | 41.84 | 42.46 | 43.39 | 41.84 | 42.46 | 43.39 | 0.94 |
| | | 27 | 37.56 | 39.60 | 40.26 | 37.56 | 39.60 | 40.26 | 0.93 |
| | | 32 | 33.58 | 37.38 | 37.89 | 33.57 | 37.37 | 37.88 | 0.92 |
| | | 37 | 29.95 | 35.80 | 36.24 | 29.94 | 35.78 | 36.22 | 0.95 |
| | Racehorse | 22 | 43.43 | 43.25 | 44.28 | 43.42 | 43.24 | 44.27 | 0.97 |
| | | 27 | 39.41 | 40.16 | 41.60 | 39.41 | 40.14 | 41.58 | 0.97 |
| | | 32 | 35.49 | 38.08 | 39.39 | 35.48 | 38.05 | 39.36 | 0.93 |
| | | 37 | 32.01 | 36.57 | 37.61 | 32.00 | 36.50 | 37.54 | 0.93 |

including class A1, class A2, class B, class C, and class D, for a total of 21 sequences. From the experimental performance, the resolution will influence the decreased time, and high-resolution sequences are more likely to have more smooth areas than the lower resolution sequences. The quantization parameter is also a factor affecting the result, in common, and the higher the QP, the higher the probability it contains more smooth areas across the frame. The experiment was only conducted with all intra configuration, and the proposed algorithm has about 10%, 11%, 7%, 5%, and 5% decoding time reductions for class A1, class A2, class B, class C, and class D, respectively, compared with the anchor software.

## Acknowledgement

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2014R1A2A1A11052210) and the Ministry of Science, ICT and Future Planning (MSIP), Korea, under the University Information Technology Research Center support program (IITP-2016-R2718-16-0014) supervised by the Institute for Information & communications Technology Promotion (IITP).

## References

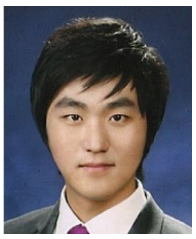
[1] ITU-T Rec. H.165 and ISO/IEC 23008-2, "High efficiency video coding," Final draft approval Jan. 2013.3 Article (CrossRef Link)

[2] W. Lim, J. Nam, D. Sim, "Scalable Multi-view Video Coding based on HEVC", IEIE Transactions on Smart Processing and Computing, vol 4, no. 6, December 2015 Article (CrossRef Link)

[3] J. Ma, D. Sim, "Early Termination of Block Vector Search for Fast Encoding of HEVC Screen Content Coding", IEIE Transactions on Smart Processing and Computing, vol.3 no. 6, December 2014 Article (CrossRef Link)
[4] T. Wiegand, G.-J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," IEEE Trans. Circuits Syst. Video Technol., vol. 13, no. 7, pp. 560-576 (2003) Article (CrossRef Link)
[5] JEM reference software Article (CrossRef Link)
[6] HEVC reference model Article (CrossRef Link)
[7] J. L. Chen, E. ALshina, Gary J. Sullivan, Jens-Rainer Ohm, J. Boyce, "Algorithm Description of Joint Exploration Test Model 3", JVET-C1001, 3rd JVET Meeting Article (CrossRef Link)
[8] X. F. Zhang, R. Q. Xiong, S. W Ma, W. Gao, "Adaptive Loop Filter with Temporal Prediction", 2012 Picture Coding Symposium Article (CrossRef Link)
[9] Y. F. Zhang, Y. Z. Tang, B. Li, "Spatial Correlation and Texture-Based Fast Adaptive Loop Filter for HEVC", 2012 International Conference on Wireless Communications and Signal Processing (WCSP) Article (CrossRef Link)
[10] M. Budagavi, V. Sze, M. H. Zhou, "HEVC ALF Decode Complexity Analysis and Reduction", 2011 18th IEEE International Conference on Image Processing Article (CrossRef Link)
[11] C. Y. Tsai, C. Y. Chen; C. M. Fu, Y. W. Huang; S. M. Lei, "One-Pass Encoding Algorithm for Adaptive Loop Filter in High-Efficiency Video Coding", 2011 Visual Communications and Image Processing (VCIP) Article (CrossRef Link)
[12] S. H. Park, K. H. Choi, G. G. Noh, Euee S. Jang, "Frame-based adaptive selection of ALF for Fast HEVC Decoding", IEEE international Symposium on Broadband Multimedia Systems and Broadcasting Article (CrossRef Link)
[13] C. Y. Tsai, C. Y. Chen, T. Yamakage, I. S. Chong, Y. W. Huang, C. M. Fu, T. Itoh, T. Watanabe, T. Chujoh, M. Karczewicz, S.M. Lei, "Adaptive Loop Filtering for Video Coding", IEEE Journal of Selected Topics in Signal Processing, VOL. 7, No. 6, DECEMBER 2013. Article (CrossRef Link)
[14] Saeed V. Vaseghi, "Advanced Digital Signal Processing and Noise Reduction, Second Edition, Chapter 6 Wiener Filter" Article (CrossRef Link)
[15] Wikipedia Wiener Filter Article (CrossRef Link)
[16] Sebastian Seung 9.29 Lecture 3: February 11, 2003 "Wiener-Hopf equation, Convolution and correlation in continuous time "Article (CrossRef Link)
[17] Alan V. Oppenheim and George C. Verghese, "Signals, Systems and Inference, Chapter 11 Wiener Filter", 2010, Pearson Article (CrossRef Link)
[18] F. Bossen, "Common test conditions and software reference configurations", JCTVC-L1100, 12th JCT-VC meeting, Geneva, CH, Jan. 2013. Article (CrossRef Link)
[19] Gisle Bjontegaard, "Calculation of average PSNR differences between RD-curves", VCEG-M33, Thirteenth Meeting: Austin, Texas, USA, 2-4 April, 2001 Article (CrossRef Link)



**Xiang Li** was born in Hunan Province, the People's Republic of China. He received a BSc from Guilin University of Technology, Guangxi Province, China, and he is now in a master's program in Kwangwoon University, Seoul, Korea. His current research interests are video scene change detection, image processing, video communications, and networking.



**Yongjo Ahn** received a BSc and MSc in Computer Engineering from Kwangwoon University, Seoul, Korea, in 2010 and 2012, respectively. He received a PhD from the same university in 2006. His research interests are high-efficiency video compression, parallel processing for video coding, and multimedia systems.

**Donggyu Sim** was born in Chungcheong Province, Korea, in 1970. He received a BSc and MSc in Electronic Engineering from Sogang University, Seoul, Korea, in 1993 and 1995, respectively. He also received a PhD from the same university in 1999. He was with the Hyundai Electronics Co., Ltd. from 1999 to 2000, where he was involved in MPEG-7 standardization. He was a senior research engineer at Varo Vision Co., Ltd., working on MPEG-4 wireless applications from 2000 to 2002. He worked for the Image Computing System Lab (ICSL) at the University of Washington as a senior research engineer, analysis and parametric video coding. He joined the Department of Computer Engineering at Kwangwoon University, Seoul, Korea, in 2005 as an Associate Professor. He was elevated to an IEEE Senior Member in 2004. His current research interests are image processing, computer vision, and video communications.