COMPUTATIONALLY EFFICIENT ADAPTIVE LOOP FILTERING DESIGN WITH DIRECTIONAL FEATURES FOR VIDEO CODING

PoLin Lai and Felix C. A. Fernandes

Dallas Technology Lab., Samsung Telecommunications America
1301 E. Lookout Dr., Richardson, TX USA 75082

ABSTRACT

To improve coding efficiency and visual quality in video coding, adaptive loop filtering (ALF) methods using directional features to classify pixels for filter adaptation have been proposed in the literature. However, the associated complexity has been identified as an obstacle for realistic hardware implementation and deployment. This paper presents an ALF design, which efficiently classifies video blocks by computing the direction and strength of local gradients only at subset of pixels. As compared to the state-of-the-art ALF approach utilizing directional features for classification, our design achieves appealing complexity-efficiency trade-off, with more than 85% reductions in computation, 75% reduction in terms of the number of pixels to be accessed, and reduces the worst-case number of filters from 15 to 8; while being able to preserve more than 85% of ALF gains in different coding structures.

Index Terms— Adaptive filter, directional filter, de-noising filter, image/video classification, directional features

1. INTRODUCTION

Adaptive loop filtering (ALF) for video coding is a technique to reduce coding artifacts introduced by hybrid video coding schemes (namely, block-based motion compensation followed by transforms and quantization). To achieve this goal, Minimum Mean-Squared Error (MMSE) Wiener filters are designed on a frame by frame basis. Filtered frames will have improved quality, and will serve as better references for temporal predictive coding. Filter coefficients are encoded and transmitted as side information. Thus, the rate-distortion tradeoff in ALF design is between the amount of side information and the improvement in picture fidelity. The key to an effective ALF scheme is identifying the underlying artifacts and designing a set of filters suitable for compensating the corresponding artifacts.

Depending on the targeted artifacts, different filter design approaches have been proposed. For specific video content exhibiting focus changes, Lai *et al.* [1] designed multiple filters by classifying macroblocks (MB) based on the type of estimated local MB-wise focus changes. For each class of blocks, a MMSE filter is designed to compensate for the associated type of focus change. For more generic denoising purposes, Karczewicz *et al.* [2] extended single filter with quadtree-based filtering on/off control by Chujoh *et al.* [3], to multiple filter ALF scheme, in which individual pixels in a frame are classified into 16 classes according to the magnitudes of sums of local Laplacian. A MMSE filter is designed for each class of pixels. While improved coding efficiency has been reported [4], the main drawback of this approach is that since only the magnitude of local Laplacian is considered, the resulting linear Wiener filters tend to be low-pass and *isotropic* in general, and hence are not suitable for reducing artifacts around sharp edges.

To introduce *directional filtering capability*, Lai *et al.* [5] and Ikai *et al.* [6] classified pixels in a frame into classes with different *directional orientations*, by computing and comparing the relative strength among multiple directional features. Higher coding efficiency and improved subjective quality are achieved, as compared to classification only using sums of local Laplacian [3]. Note that in order to correctly apply filters to the corresponding pixels without sending the localized filter indices, the decoder needs to perform the same classification process as done at the encoder side. Thus for practical implementations, it is highly preferred that the feature computation and classification to be simple. It has been proposed to perform classification on block-basis [6][7][8]: Blocks (rather than individual pixels) in a video frame are classified based on multiple *block-wise directional features*. This reduces the complexity at decoder, since it only has to apply one filter to all pixels within a given block as compared to pixel-based classification where potentially the decoder might need to apply different filters to consecutive pixels. Among them, the method by Chong *et al.* [8] considers both directional features and sums of local Laplacian magnitudes, to classify blocks for filter design, leading to the highest coding efficiency. It was adopted into High Efficiency Video Coding (HEVC) [9], the ongoing standardization for a new video coding standard, held by the Joint Collaborative Team on Video Coding (JCT-VC).

However, due to the calculations of sums of Laplacian magnitudes, the complexity to compute features remains too high for realistic hardware implementations [10]. To address this issue, Lai *et al.* [7][11] proposed to compute directional features only at subset of pixels and also avoid computing the sums of Laplacian. As compared to the method by Chong *et al.* [8], the later work [11] reduces the operations to compute features by more than 80%, while maintaining 90% of the achievable gains. It has been adopted into HEVC especially due to the endorsement from companies

producing video codec hardware. In this paper, we will further reduce the computation and also reduce the access area required to compute directional features.

Besides the complexity associated with feature computation, Chong *et al.* [8] classifies blocks into 15 classes based on directional and sums of Laplacian features. Using 15 filters increases side information (filter coefficients) and might consequently reduce the efficiency of ALF. One possible approach to reduce the number of filters [12] is to conduct rate-distortion tests to merge different class-labels and redesign filters. Such a process will increase encoding complexity. Thus, there is also a need for simpler ALF processing with fewer filters and less need for the associated filter-redesign process.

In this paper, we present a computationally efficient ALF classification scheme using the direction and strength of local gradients, which reduces computation complexity, access area, and the number of filters, while preserving most of the coding gains. The remainder of the paper is organized as follows: Section 2 describes the state-of-the-art ALF classification which utilizes both directional and Laplacian features [8]. The proposed computationally efficient classification for ALF is presented in Section 3. We conduct complexity analysis for both methods. Simulation results are provided in Section 4, with conclusions in Section 5.

2. FILTER ADAPTATION USING DIRECTIONAL AND LAPLACIAN FEATURES

In the state-of-the-art ALF approach [8], to classify a 4×4 block with pixels $\{X(i,j) \mid i=0,1,2,3;\ j=0,1,2,3\}$, directional features H, V and sums of Laplacian feature L_S are computed at *every* pixel $X(i,j)$ as follows:

$$H(i,j) = |\, X(i,j)<<1 - X(i,j-1) - X(i,j+1)\,| \quad (1)$$

$$V(i,j) = |\, X(i,j)<<1 - X(i-1,j) - X(i+1,j)\,| \quad (2)$$

$$L(i,j) = H(i,j) + V(i,j) \quad (3)$$

$$L_S(i,j) = \sum_{n=-1,0,1} \sum_{m=-1,0,1} L(i+m,j+n) \quad (4)$$

As shown above, for each pixel, H and V measure the horizontal and vertical variations respectively (directional features); while L_S measures the sum of Laplacian within a 3×3 window. Then, the block-wise features are computed as:

$$H_B = \sum_{i=0,1,2,3} \sum_{j=0,1,2,3} H(i,j) \quad (5)$$

$$V_B = \sum_{i=0,1,2,3} \sum_{j=0,1,2,3} V(i,j) \quad (6)$$

$$L_B = \sum_{i=0,1,2,3} \sum_{j=0,1,2,3} L_S(i,j) >> \quad (7)$$

Finally, the class label C_B of the 4×4 block B is determined by comparing the relative strength between directional features H_B and V_B (*dir*), and the *normalized* Laplacian magnitude average L_B (*norMag)*,as below:

- $$dir = \begin{cases} 1, & \text{if } V_B > 2\times H_B \\ 2, & \text{if } H_B > 2\times V_B \\ 0, & \text{otherwise} \end{cases} \quad (8)$$
- $$norMag = \max\{\, 15, (L_B\times 114) >> 11 \,\} \quad (9)$$

C_B = **classTab** [*dir*][*norMag*] as Table 1 below.

From (8), the block-based classification produces 3 directional classes for filter adaptation: *dir* = 1 and *dir* = 2 correspond to blocks with strongly predominant vertical and horizontal variations, respectively; while for blocks with *dir* = 0, variation in either direction is not much stronger than the variation in the other direction. Then within each directional class, there are 5 variance (magnitude) levels, determined based on the normalized Laplacian magnitude *norMag*. Note that *norMag* saturates at 15 when L_B exceeds certain value. For each row in Table 1, as *norMag* increases, the class labels cover larger L_B ranges. This was designed to take into account the fact that, in typical video content, there are more blocks with smaller Laplacian values L_B (blocks with small pixel value variation) and much fewer blocks with very large L_B (large pixel value variation) [6].

Table 1: **classTab** to classify blocks into 15 classes

	norMag							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
***dir* 0**	0	1	2	2	2	3	3	3
***dir* 1**	5	6	7	7	7	8	8	8
***dir* 2**	10	11	12	12	12	13	13	13

norMag							
8	**9**	**10**	**11**	**12**	**13**	**14**	**15**
3	3	4	4	4	4	4	4
8	8	9	9	9	9	9	9
13	13	14	14	14	14	14	14

After classification, for all blocks in each class, an MMSE filter will be designed.

2.1. Complexity Analysis for the Classification in [8]

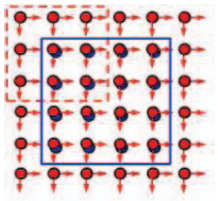

Fig. 1 Feature computation in [8] for a 4×4 block, and the weighting matrix applied to the per pixel Laplacian *L*

In this subsection, we provide analysis for the complexity of the classification method by Chong *et al.* [8]. Fig. 1 illustrates the computation for classifying a 4×4 block, shown within the solid-lined rectangle. At each pixel, obtaining L_S in (4) requires 9 Laplacians L. The example for computing $L_S(0,0)$ is shown in Fig. 1, where 9 Laplacians are computed at pixels within the dashed 3×3 rectangle. Therefore, obtaining L_B requires 36 Laplacians L as depicted in Fig. 1, and accessing 8×8 pixels (Laplacians need one additional line of pixels on top / bottom / left / right). From (4) and (7), the weights applied to these 36 Laplacians L are listed as the weighting matrix in Fig. 1. In an optimized implementation, this 2D weighted summation is decomposed into separable 1D summations in vertical and horizontal directions, each with weights {1,2,3,3,2,1}:

First for each row, compute the sum $(L(i,-1)+L(i,4)) + (L(i,0)+L(i,3))<<1 + (L(i,1)+L(i,2))\times 3$. The multiplication by 3 can be implemented using a shift operation and an addition, leading to a total of 6 additions to compute the sum for each row (and there are 6 rows). Then sum over these 6

values again with weights {1,2,3,3,2,1}, which requires another 6 additions. Thus, from $L(i,j)$, we can obtain L_B with a total of 42 additions.

Thus, the total number of operations to compute the class label C_B for a 4×4 block is:

- H and V in (1)(2): 16×4 additions, 16×2 abs values
- L in (3): 16 additions
- H_B and V_B in (5)(6): Another 15 additions each
- L_B via L_S as discussed above: 42 additions

→ 152 additions, and 32 absolute values (Note that shifting operations are not counted, as it is common practice to not include them especially when assessing hardware complexity.)

3. COMPUTATIONALLY EFFICIENT FILTER ADAPTATION SCHEME

To achieve a filter adaptation scheme with reduced complexity and number of filters, while preserving the directional filtering capability, we propose the following changes to the classification process:

a) Feature computation only at subset of pixels: Since the features are for classifying blocks, it is not necessary to compute features at every pixel. As an example, we use only pixels uniformly subsampled by 2.

b) Avoid computing the sums Laplacians L_S: For each pixel being used to compute features, instead of computing 9 Laplacians within a 3×3 window, the variance is simply measured as the sum of H and V. Thus, the corresponding normalization factor to obtain *norMag* from Laplacian magnitude average L_B, has to be changed from 114 in (9) to 1024. (The original value 114 in the method [8] was computed from the normalization 1024 / 9 ≈ 114.)

c) Simple gradients as directional features: We modified the features H and V from [-1,2,-1] in (1)(2) to gradients [1,-1]. Although operator [-1,2,-1] can detect directional edges, it may not be able to detect gradual value changes along certain direction. From signal processing perspective, the features [1,-1] capture *directional gradients*, which can be more effective in capturing directional textures (directional gradual change, and direction edges). Complexity-wise, this eliminates half of the subtractions at each pixel used for computing features. Combining with a) and b), this also leads to a reduced access area of exactly the 4×4 pixels within B as depicted in Fig. 2.

d) Two directional classes, with four variance levels: The classification in Section 2 creates three directional classes according to the relative strength of ratio 2 between H_B and V_B. However, since there is another dimension *norMag* for classification which already measures the strength of variance within each directional classes, we propose a coarser classification using directional and variance features: Two directional classes are determined by directly comparing directional gradients H_B and V_B: dir = 1 corresponds to blocks with larger vertical gradient than horizontal gradient ($H_B > V_B$), while dir = 0 correspond to the other type of blocks ($H_B < V_B$). Then for each directional class, we construct 4 magnitude levels by stretching out the label ranges row-wise in Table I to better classify *norMag*. Thus, the total number of filters is reduced from 15 to 8, before applying any optional class-label-merging process such as [12]. This significantly reduces the worst-case number of filters.

Our ALF classification with directional features can be summarized as below:

For a 4×4 block, compute directional features only at pixels subsampled by 2: $\{X(i,j) \mid i=0,2; j=0,2\}$, using simple directional gradients:

$$H(i,j) = | X(i,j) - X(i,j+1) | \quad (10)$$
$$V(i,j) = |X(i,j) - X(i+1,j) | \quad (11)$$

Then the block-wise features are computed as:

$$H_B = \sum_{i=0,2} \sum_{j=0,2} H(i,j) \quad (12)$$
$$V_B = \sum_{i=0,2} \sum_{j=0,2} V(i,j) \quad (13)$$
$$L_B = (H_B + V_B) >> 1 \quad (14)$$

The class label C_B of block B is determined as the Table 2 **classTab2** below, C_B = **classTab2** [*dir*][*norMag*]:

- dir = 1, if $H_B > V_B$
  0, otherwise (15)
- $norMag$ = max{ 15, (L_B×1024) >> 11 } (16)

Table 2. **classTab2** to classify blocks into 8 classes

	norMag							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
***dir* 0**	0	0	1	1	1	1	1	2
***dir* 1**	4	4	5	5	5	5	5	6

norMag							
8	**9**	**10**	**11**	**12**	**13**	**14**	**15**
2	2	2	2	2	2	2	3
6	6	6	6	6	6	6	7

3.1. Complexity of the Proposed Classification

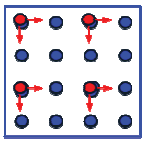

Fig.2 Proposed feature computation for a 4×4 block

Thanks to the feature computation only at subset of pixels, the elimination of sum of Laplacians within 3×3 windows, and the use of gradient features, the complexity of the proposed classification is significantly reduced. For a 4×4 block, the operations are as the following:

- H and V in (10)(11) at 4 pixel: 4×2 adds, 4×2 abs values
- H_B and V_B in (12)(13): Another 3 additions each
- L_B in (14): Simply 1 additions

→ 15 additions, 8 absolute values. This presents a drastic reduction as compared to the operations in Section 2.

4. SIMULATION RESULTS

We implemented our classification for filter adaptation in HEVC reference software HM4.0. Simulations were conducted with QP 22, 27, 32, and 37 to obtain different rate-points. Four different coding structures were tested: All Intra, Random Access (hierarchical B pictures), Low Delay B (IPPP pictures with 2 references lists) and Low Delay P (IPPP pictures with 1 reference list). We measured the coding efficiency gains provided by ALF using the classification method by Chong *et al.* [8] and using our scheme; in terms of BD-rate savings (negative numbers represent coding efficiency gains) [13]. The complete comparison is summarized in Table 3.

It can be seen that, with significant complexity reductions in terms of computation, access area, and number of filters, our proposed classification scheme is able to preserve 85% of the ALF gains achieved by Chong *et al.* [8] for Intra coding (1.7% versus 2.0%), and preserve more than 90% of its gains for Random Access, Low Delay B and Low Delay P coding structures.

Table 3. Complexity and performance comparison

		ALF using classification method in [8]	ALF using proposed classification
Classification Complexity for 4×4 block	***Additions***	***152***	***15***
	Abs. Values	***32***	***8***
	Access area	***8×8***	***4×4***
Number of filters		***15***	***8***
Gains for All Intra	4K	-3.2%	-2.9%
	1080P	-1.9%	-1.7%
	WVGA	-1.6%	-1.2%
	WQVGA	-0.7%	-0.6%
	720P	-2.8%	-2.3%
	Average	***-2.0%***	***-1.7%***
Gains for Random Access	4K	-5.4%	-5.1%
	1080P	-3.9%	-3.5%
	WVGA	-2.6%	-2.3%
	WQVGA	-2.5%	-2.4%
	720P	N/A	N/A
	Average	***-3.6%***	***-3.3%***
Gains for Low Delay B	4K	N/A	N/A
	1080P	-3.4%	-3.0%
	WVGA	-3.1%	-2.7%
	WQVGA	-1.8%	-1.7%
	720P	-4.3%	-4.0%
	Average	***-3.1%***	***-2.8%***
Gains for Low Delay P	4K	N/A	N/A
	1080P	-7.3%	-6.7%
	WVGA	-3.5%	-3.0%
	WQVGA	-1.3%	-1.1%
	720P	-9.2%	-8.7%
	Average	***-5.2%***	***-4.7%***

5. CONCLUSIONS

In this paper, we present efficient classification method for ALF filter adaptation, which measures the direction and strength of local gradients only at subset of pixels. Compared to the state-of-the-art ALF approach utilizing directional features, the proposed scheme reduces computation by more than 85%, reduces the number of pixels required to be accessed by 75% (from 8×8 to 4×4), and reduces the worst-case number of filters from 15 to 8, while being able to preserve more than 85% of the ALF gains in coding structures such as intra coding, hierarchical B, and low delay with IPPP. The complexity-efficiency trade-off is attractive to hardware implementations. As a result, part of the scheme presented in this paper has already been adopted into HEVC, the standardization for a new video coding standard.

6. REFERENCES

[1] P. Lai, Y. Su, P. Yin, C. Gomila, and A. Ortega, "Adaptive Filtering for Video coding with Focus Change", in *Proc. IEEE ICASSP 2007*, Honolulu, HI, USA, pp. I.661-664, Apr. 2007

[2] W.-J. Chien and M. Karczewicz, "Adaptive Filter based on Combination of Sum-Modified Laplacian Filter Indexing and Quartree Pratitioning", ITU-T Q.6/SG16 VCEG-AL27, London, UK, Jul. 2009

[3] T. Chujoh, N. Wada, T. Watanabe, G. Yasuda, and T. Yamakage, "Specification and experimental results of quadtree-based adaptive loop filter", ITU-SG16 Q6 VCEG-AK22, Apr. 2009

[4] T. Yamakage, T. Chujoh, and T. Watanabe, "Comparison of Loop / Post Filtering for In-loop and Post Processing filtering AHG", JCTVC-C085, Guangzhou, China, Oct. 2010

[5] P. Lai and F. C. A. Fernandes, "Loop Filter with Directional Similarity Mapping (DSM)", JCTVC-D221, Daegu, Korea, Jan. 2011

[6] T. Ikai and Y. Yasugi, "Region-based Adaptive Loop Filter using Two-dimensional Feature", JCTVC-D116, Daegu, Korea, Jan. 2011

[7] P. Lai and F. C. A. Fernandes, "Loop Filtering with Directional Features", JCTVC-E288, Geneva, Switzerland, Mar. 2011

[8] I. S. Chong, M. Karczewicz, C.-Y. Chen, C.-M. Fu, C.-Y. Tsai, Y.-W. Huang, S. Lei, T. Yamakage, T. Chujoh, and T. Watanabe, "CE8 Subtest 2: Block based Adaptive Loop Filter (ALF)", JCTVC-E323, Geneva, Switzerland, Mar. 2011

[9] "WD4: Working Draft 4 of High-Efficiency Video Coding", JCTVC-F803, Torino, Italy, Jul. 2011

[10] T. Hellman, "ALF Complexity Analysis", JCTVC-F342, Torino, Italy, Jul. 2011

[11] P. Lai, F. C. A. Fernandes, and I.-K. Kim, "CE8 Subtest 1: Block-based Filter Adaptation with Features on Subset of Pixels", JCTVC-F301, Torino, Italy, Jul. 2011

[12] T. Ikai, Y. Yasugi, T. Yamazaki, M. Karczewicz, and I. S. Chong, "CE8.1:Block based Adaptive Loop Filter with flexible syntax and additional BA mode", JCTVC-F384, Torino, Italy, Jul. 2011

[13] G. Bjontegaard, "Calculation of Average PSNR Differences between RD curves," ITU-T SG16/Q6, VCEG-M33, Apr. 2001