Deblocking Filter Algorithm with Low Complexity for H.264 Video Coding

Jung-Ah Choi and Yo-Sung Ho

Gwangju Institute of Science and Technology (GIST)
261 Cheomdan-gwagiro, Buk-gu, Gwangju, 500-712, Korea
{jachoi,hoyo}@gist.ac.kr

Abstract. In H.264, block-based discrete cosine transform (DCT) and block-based motion compensated prediction are used to reduce both spatial and temporal redundancies. Due to the block-based coding, discontinuities at block boundaries, referred to blocking artifacts are created. To reduce these blocking artifacts, H.264 employs a deblocking filter. However, it causes a significant amount of complexity; therefore, the deblocking filter occupies one-third of the computational complexity of the decoder. In this paper, we propose a deblocking filtering algorithm with low complexity. Using boundary strength (BS) of the first Line-of-Pixel (LOP), we determine BS of successive LOP in advance. Then, we apply deblocking filters including newly designed filters. Experimental results show that the proposed algorithm provides an average computations reduction of 73.45 % in the BS decision. In the filter implementation, it reduces an average 57.52 % of additions, 100 % of multiplications, and 5.66 % of shift operations compared to the deblocking filter in H.264 with comparable objective quality.

Keywords: H.264, Deblocking Filter.

1 Introduction

The H.264 video coding standard was developed through the Joint Video Team (JVT) from the ITU-T Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group (MPEG) standardization. H.264 is the most advanced development for video coding [1].

H.264 provides better coding efficiency compared with the previous standards, such as MPEG-4 and H.263, due to various techniques. The remarkable techniques of H.264 include variable block motion compensation, multiple reference images, 1/4-pixel motion vector accuracy, and in-loop deblocking filter. Among these various techniques, the in-loop deblocking filter has a significant impact on the perception quality of video [2] [3].

Deblocking filter plays an important role in today's video and image coding applications. Since today's video coding uses block DCT-based coding techniques, it brings about visible blocking artifacts. Blocking artifacts can be defined by discontinuities occurred at each block boundary.

Y.-M.R. Huang et al. (Eds.): PCM 2008, LNCS 5353, pp. 138–147, 2008.

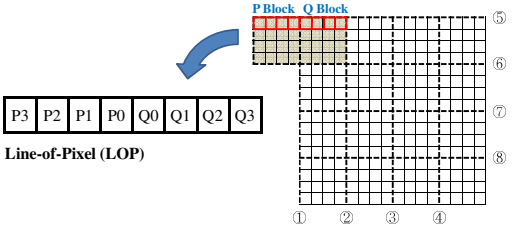
In the typical block-based coding, it is necessary to decrease such blocking artifacts at the boundaries of the macroblock. In H.264, the deblocking filter is applied in both encoder and decoder.

For each filter operations, eight pixels (P3, P2, P1, P0, Q0, Q1, Q2, Q3) on both side of the boundary are inputted to deblocking filter, and some of the 8 pixels are modified after filtering according to boundary strength (BS) and other thresholds. eight pixels in one line of block can be grouped as Line-of-Pixel (LOP), as shown in Fig. 1. In the deblocking filter in H.264, we check BS in every LOP of the macroblock. Then, the filter smoothes block edges, improving the appearance of decoded frames.

However, the deblocking filter has a drawback that is large computational complexity. Especially, the deblocking process can consume about 30 - 35 % of the decoding computations [4]. Also, filter equations are too complex, although it gives good filtering results.

At the decoder, the complexity problem is most important, because the complexity at the decoder is directly related to users. For deploying on a commercial scale, the decoder complexity is major problem. The deblocking filter is one of the main parts, which causes the decoder complexity. Therefore, if we create a simple and efficient deblocking filter, it is profitable to solve the complexity problem.

In this paper, we propose a new deblocking filtering algorithm to reduce the computational complexity of the entire process. The H.264 deblocking filter plays a lot of computations to determine BS at each LOP. Also, the filter equations for high BS are too complex. In the proposed algorithm, using BS of the first LOP (*BSFLOP*) of each block boundary, we determine BS of successive LOP (*BSSLOP*) in advance. Then, we apply deblocking filters including newly designed filters with low computational complexity for high BS.



(a) Vertical Boundaries (b) Horizontal Boundaries

Fig. 1. Sequential Order of Vertical and Horizontal Boundaries in a Macroblock

2 Overview of Deblocking Filter in H.264

In this section, we briefly describe a deblocking filter in H.264. In H.264, the deblocking filter is used to decrease blocking artifact at block boundaries. The filtering is done first from left to right vertically and then from top to bottom on the horizontal boundaries. Fig. 1 represents vertical and horizontal boundaries in one macroblock. Each square stands for a block of 4×4 pixels.

In order to apply a filter to each macroblock, we use filtered pixels at the top and on the left of the current macroblock. Luminance and chrominance components are separately processed. In H.264, the deblocking filtering process consists of three operations: BS decision to determine the filter strength, mode decision to select the suitable filter, and filter implementation [5].

The deblocking filter in H.264 achieves substantial objective and subjective quality improvements. However, the H.264 deblocking filter has a drawback that the computational complexity of the process is too high. In the BS decision, we check several conditions in every LOP to determine the suitable BS. In the filter implementation, the number of operations: additions, multiplications, and shift operations in filter equations are too much. These are main factors of the computational complexity. Therefore, a new deblocking filtering algorithm with low complexity is required.

3 Low Complex Deblocking Filter Algorithm

3.1 Fast BS Decision

BS decision in H.264 examines each LOP in the macroblock. It checks several conditions for determining a suitable BS. However, in many cases, pixels in the same boundaries have similar amount of blocking artifacts; therefore, they might have similar BS. Thus, we can assume that BS_{FLOP} is equal to BS_{SLOP} in the horizontal or vertical boundary. To confirm the assumption, we checked BS_{SLOP} for each BS_{FLOP} value.

Table 1 shows the statistical results of BS_{SLOP} distribution for various sequences and various quantization parameters (QP). We use ten test sequences (Foreman, Hall Monitor, Mother and Daughter, Container, Coastguard, Stefan, News, Table Tennis, Salesman, and Carphone) with 100 frames. The coding structure is IPPP…P.

Table 1. BS_{SLOP} distribution for various sequences (%)

Conditions	$BS_{SLOP} = 0$	$BS_{SLOP} = 1$	$BS_{SLOP} = 2$	$BS_{SLOP} = 3$	$BS_{SLOP} = 4$
$BS_{FLOP} = 0$	97.02	0.82	2.16	0	0
$BS_{FLOP} = 1$	6.92	88.79	4.29	0	0
$BS_{FLOP} = 2$	38.60	7.95	53.45	0	0
$BS_{FLOP} = 3$	0	0	0	100	0
$BS_{FLOP} = 4$	0	0	0	0	100

From the experiment, we can observe that for BS equal to zero, three, and four, our assumption is reasonable; however, for BS equal to one and two, it is not sufficient value to unify BS into one value. Hence, if BS_{FLOP} is zero, three, or four, we can easily determine BS_{SLOP} without any computations. In the conventional BS decision, deblocking filter checks three conditionals for BS equal to zero and two conditionals for BS equal to three or four. Moreover, it is performed at each LOP. Therefore, if we determine BS using above BS_{SLOP} distribution, we can reduce many conditionals for BS decision. In the proposed algorithm, when BS_{FLOP} is equal to zero, we determine BS_{SLOP} is zero. Also, when BS_{FLOP} is equal to three and four, we set BS_{SLOP} as three and four, respectively.

3.2 Deblocking Filter with Low Complexity

The deblocking filter in H.264 has only two filtering modes: normal mode and special mode. If BS is from one to three, normal mode is selected. Otherwise, special mode is chosen as a filtering mode. However, characteristics of BS equal to 1 or 2 and BS equal to 3 is not same. Thus, we need to separate these two cases. Another drawback of the H.264 deblocking filter is an amount of complexity of filtering. Despite of good performance, cost of computational complexity is too much. Hence, we need to propose a simple deblocking filter. In the proposed algorithm we design two deblocking filters with low complexity for high BS.

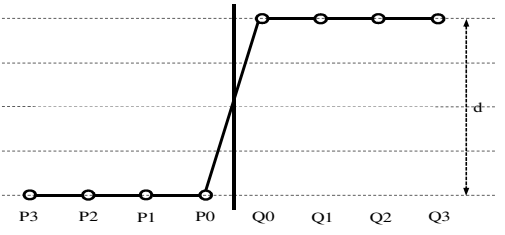


Fig. 2. Block Boundary after Decoding

One dimensional view of block boundaries before and after deblocking filtering is shown in Fig. 2 and 3. This kind of filtering requires less amount of computation compared to N-tap low pass filtering in the conventional deblocking filter. The proposed algorithm requires very simple control mechanism for applying filtering in comparison to other well-known algorithms.

1) Filter for BS equal to 1 or 2: If BS value is less than 2 (BS = 1 or 2), we apply normal mode filter for low BS in the deblocking filter in H.264 [6]:

$$P0' = P0 + \Delta_0 \tag{1}$$

$$Q0' = Q0 - \Delta_0 \tag{2}$$

where $\Delta_0 = (4(Q0 - P0) + (P1 - Q1) + 4) >> 3$.

2) Filter for BS equal to 3: We describe strong mode filtering for luminance. This algorithm is a modification to the algorithm proposed by Ramkishor [7]. The filtering process is shown in Fig. 3 (a).

First, we calculate the size of discontinuity, d. In this mode, four pixels around the block boundary are filtered. Thus, we can predict that a smoothing line is the connection of P2 and Q2. In order to match pixel values with the predicted smoothing line, we can calculate pixel values as

$$P0' = P0 + (d >> 1) - \Delta \tag{3}$$

$$P1' = P1 + (d >> 2) + (\Delta >> 1) \tag{4}$$

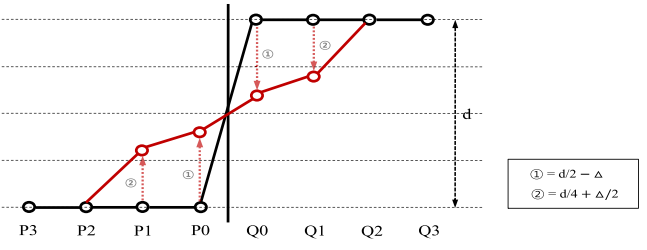
$$Q0' = Q0 - (d >> 1) + \Delta \tag{5}$$

$$Q1' = Q1 - (d >> 2) - (\Delta >> 1) \tag{6}$$

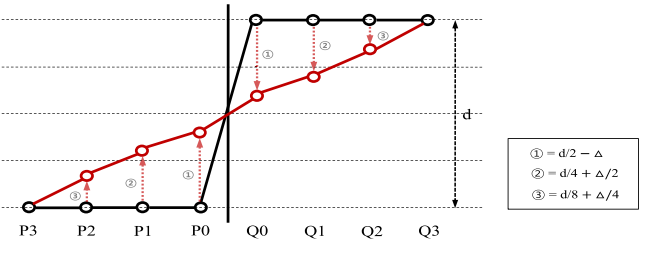
The amount of modification that will be applied to each of the edge samples is obtained as

$$\Delta = d/5 \tag{7}$$

The algorithm is applied to all blocks.



(a) Filter for BS equal to 3



(b) Filter for BS equal to 4

Fig. 3. Block Boundary after Deblocking Filtering

3) Filter for BS equal to 4: In strong mode filter, only four pixels around the block boundary are filtered. This filter operation avoids blurring the regions with high spatial details, but restricts the filter effect for regions with strong blocking artifact. The strength of filtering can be improved if the filter length becomes long. In case of BS equal to 4, the smoothing line is changed to the connection of P3 and Q3. In addition to the result of strong mode filter, we modify two more pixels as

$$P2' = P2 + (d >> 3) + (\Delta >> 2) \tag{8}$$

$$Q2' = Q2 - (d >> 3) - (\Delta >> 2) \tag{9}$$

and the △ value is changed as

$$\Delta = d/7 \tag{10}$$

One dimensional view of block boundaries after deblocking filtering is shown in Fig. 3 (b). This kind of filtering requires less amount of computation compared to N-tap low pass filtering in the conventional deblocking filter. Table 2 shows the number of operations for one LOP filtering. Since we design filters for BS equal to 3 and 4,

Table 2. The Number of Operations during One Filtering (BS = 3 and 4)

Operations	JM 11.0		Proposed Filter	
	BS = 3	BS = 4	BS = 3	BS = 4
Additions	20	28	8	12
Multiplications	4	10	0	0
Shift Operations	6	6	4	5

we compare the number of operations in these BS values. As seen in Table 3, the proposed algorithm reduces the number of operations much. Note that in the filter equation of the proposed algorithm, there is no multiplication. The proposed algorithm requires very simple control mechanism for applying filtering in comparison to other well-known algorithms.

3.3 Procedure of the Proposed Algorithm

Figure 4 shows the flowchart of the proposed deblocking filter algorithm. First, we apply the fast BS decision, and determine BS equal to zero, three, and four without any computations. According to determined BS, we select suitable filter. For high BS such as BS equal to 3 and 4, we apply newly designed deblocking filter. For low BS, we apply the conventional filter.

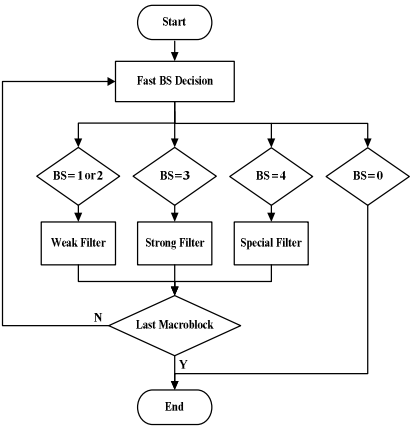


Fig. 4. Flowchart of the Proposed Algorithm

4 Simulation Results

The proposed deblocking algorithm is implemented on JM 11.0 [8]. We have tested four QCIF (176×144) video sequences (Foreman, Stefan, Coastguard, and Hall Monitor). For each sequence, 100 frames are encoded. The frame rate is 30 fps. CAVLC is adopted as the entropy coding method. Experiments were conducted for four quantization parameters: QP = 28, 32, 36, and 40.

To evaluate the complexity of the algorithm, we calculate the number of operations N_{OP} using following equation.

$$N_{OP} = N_{Fr} \times N_{MB} \times N_{LOP} \times (8 - N_{FBS}) \tag{11}$$

$$N_{FBS} = N_{BS=0} + N_{BS=3} + N_{BS=4} \tag{12}$$

where N_{Fr} is the number of encoded frames, N_{MB} is the number of macroblocks in one frame, N_{LOP} is the number of LOP in one boundary, and N_{FBS} is the number of conducted fast BS decisions in one macroblock. Since JM 11.0 does not use fast BS decision, N_{FBS} is zero. However, in the proposed method, BS of many LOPs is determined by fast BS decision, and we can reduce actual BS selection operations. In the proposed algorithm, N_{FBS} is calculated by addition of the number of BS equal to 0, 3, and 4. Table 3 shows the computational complexity comparison in BS decision. They are relative to results by the H.264 standard. The number of reduced N_{OP} is calculated using following equation.

$$\Delta N_{OP} = \frac{N_{OP}(proposed) - N_{OP}(reference)}{N_{OP}(reference)} \times 100(\%) \tag{13}$$

Table 3. Computational Complexity Comparison of BS Decision

Test Sequence	QP	H.264	Proposed	ΔN_{OP} (%)
Coastguard	28	1267200	732705	-42.18
	32	1267200	481560	-62.00
	36	1267200	322740	-74.53
	40	1267200	241095	-80.97
Foreman	28	1267200	366960	-71.04
	32	1267200	272055	-78.53
	36	1267200	219420	-82.68
	40	1267200	188085	-85.16
News	28	1267200	366960	-82.65
	32	1267200	272055	-85.43
	36	1267200	219420	-87.61
	40	1267200	188085	-89.01
Stefan	28	1267200	366960	-44.37
	32	1267200	272055	-60.65
	36	1267200	219420	-70.97
	40	1267200	188085	-77.34

In Table 4, average complexity comparison between the H.264 deblocking filter and the proposed deblocking filter is represented. We checked the number of additions, multiplications, and shift operations. It is easy to observe that the proposed deblocking filter needs small operations than that of H.264.

Table 4. Computational Complexity Comparison of Filter Implementation

Test Sequence	Operations	H.264	Proposed	ΔN_{OP} (%)
Coast-guard	Additions	332272	141392	-57.45
	Multiplications	113112	0	-100
	Shifts	74232	70696	-4.76
Foreman	Additions	208800	87488	-58.10
	Multiplications	63584	0	-100
	Shifts	50736	43744	-13.78
News	Additions	149136	63728	-57.27
	Multiplications	52232	0	-100
	Shifts	32520	31864	-2.02
Stefan	Additions	198432	84784	-57.27
	Multiplications	69448	0	-100
	Shifts	43296	42392	-2.09

To evaluate the objective quality of the decoded videos, we used the average peak signal-to-noise ratio (PSNR) and average bit rate. It must be noted that positive values indicate increments, and negative values indicate decrements.

$$\Delta PSNR = PSNR(proposed) - PSNR(reference)\ (dB) \tag{14}$$

$$\Delta Bitrate = \frac{Bitrate(proposed) - Bitrate(reference)}{Bitrate(reference)} \times 100\,(\%) \tag{15}$$

Table 5. Performance Comparion

Test Sequence	QP	H.264		Proposed		ΔPSNR (dB)	ΔBitrate (%)
		PSNR (dB)	Bitrate (kbps)	PSNR (dB)	Bitrate (kbps)		
Coastguard	28	34.20	311.18	34.20	310.66	0	-0.17
	32	31.17	155.62	31.19	155.42	0.02	-0.13
	36	28.67	77.60	28.67	78.07	0	0.61
	40	26.43	41.14	26.45	41.62	0.02	1.17
Foreman	28	36.50	122.32	36.52	123.43	0.02	0.91
	32	33.80	74.01	33.82	74.44	0.02	0.58
	36	31.19	46.34	31.19	46.24	0	-0.22
	40	28.38	28.23	28.41	28.39	0.03	0.57
News	28	36.81	72.15	36.81	72.22	0	0.10
	32	33.74	43.47	33.73	43.63	-0.01	0.37
	36	30.81	25.74	30.81	25.91	0	0.66
	40	30.78	25.13	30.78	25.21	0	0.32
Stefan	28	34.51	408.06	34.50	407.36	-0.01	-0.17
	32	30.95	221.95	30.97	221.63	0.02	-0.14
	36	27.81	119.01	27.81	118.95	0	-0.05
	40	24.86	65.65	24.87	65.43	0.01	-0.34

From Tables 3 to 5, we can observe that the proposed method provides significant complexity saving at the cost of negligible loss in PSNR values and a small increment in bit rate. Fig. 5 illustrates rate-distortion (R-D) curves for each test sequence. From Fig. 10, we note that the proposed method provides a similar R-D performance to the H.264 standard.

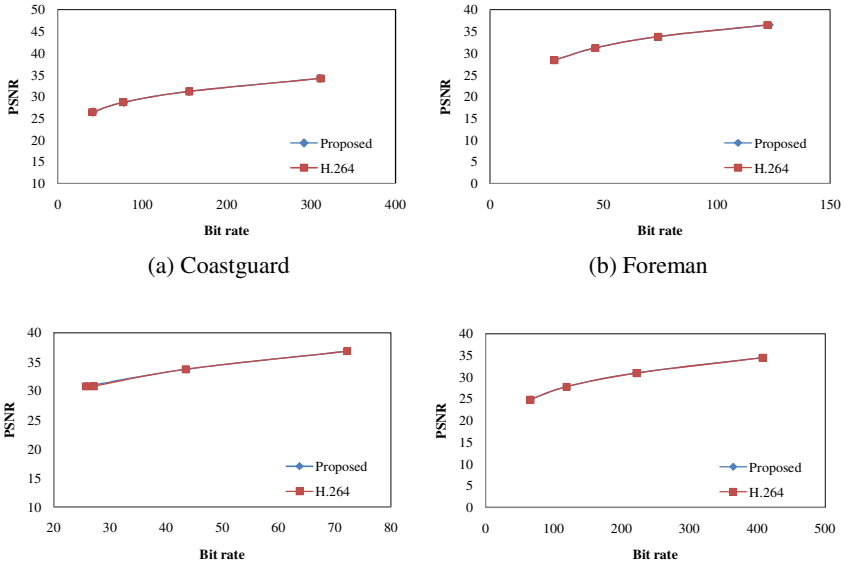


Fig. 5. Rate-distortion Curves

Fig. 6. Decoded quality comparison of deblocking filter for the Foreman sequence

Figure 6 represents the decoded images using JM 11.0 and the proposed method, respectively. The experiment was conducted with QP equal to 40. Even though a few amplitude edges are blurred, the proposed approach shows similar decoding results as the JM 11.0 deblocking filer. Therefore, we verified that the proposed deblocking filter is much better than the H.264 deblocking filter in terms of computational complexity without any major quality degradation.

5 Conclusions

In this paper, a deblocking filtering algorithm with low complexity was presented. To reduce the unnecessary BS decision, we used BS_{FLOP}. Since LOP in the same block boundary has similar characteristics, amount of blocking artifacts in this region is also similar and they have similar BS. Thus, using BS_{FLOP}, we can determine BS_{SLOP} without any computations. Then, we apply deblocking filters at each boundary according to the selected BS. For BS equal to 3 and 4, we apply newly designed filters, which is much simpler than the conventional filter. Experimental results showed that the proposed deblocking filtering algorithm reduces the significant computational complexity during BS decision and filter implementation, with a slight PSNR drop and a negligible bitrate increase. About 73.45 % of computations for BS decision is reduced and 57.52 % of additions, 100 % of multiplications, and 5.66 % of shift operations are reduced, compared with the H.264 standard.

Acknowledgements

This work was supported in part by ITRC through RBRC at GIST (IITA-2008-C1090-0801-0017).

References

1. Joint Video Team of ITU-T and ISO/IEC JTC 1, Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC), Doc. JVT-G050 (March 2003)
2. Chang, S.C., Peng, W.H., Wang, S.H., Chiang, T.: A Platform Based Bus-interleaved Architecture for De-blocking Filter in H.264/MPEG-4 AVC. IEEE Transactions on Consumer Electronics 51, 249–255 (2005)
3. Ostermann, J., Bormans, J., List, P., Marpe, D., Narroschke, M., Pereira, F., Stockhalmmer, T., Wedi, T.: Video Coding with H.264/AVC: tools, performance, and complexity. IEEE Circuits and Systems Magazine 4 (2004)
4. Lou, J., Jagmohan, A., He, D., Lu, L., Sun, M.: Statistical Analysis based H.264 high profile deblocking speed up. In: IEEE International Symposium on Circuits and Systems, pp. 3143–3146 (May 2007)
5. Richardson, I.E.G.: H.264 and MPEG-4 Video Compression_Video Coding for Next-generation Multimedia. Wiley, Chichester (2003)
6. Wiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A.: Overview of the H.264/AVC video coding standard. IEEE Transactions on Circuits and Systems for Video Technology 13(7), 560–576 (2003)
7. Ramkishor, K., Karandikar, P.: A simple and efficient deblocking algorithm for low bit-rate video coding. In: IEEE Inernational Symposium on Consumer Electronics (December 2000)
8. JVT Reference Software Version 11.0, `http://iphome.hhi.de//suehring/tml/download/jm_old/jm11.0.zip`