

A Novel Deblocking Filter Algorithm In H.264 for Real Time Implementation

Yuan Li, Ning Han, Chen Chen

Department of Automation, School of Technology
Beijing Forestry University
Beijing China 100083

e-mail: bjfu.yuan.li@gmail.cn, hn217@bjfu.edu.cn, chen_chen09@yahoo.cn

Abstract—Due to the limitation of computing complexity, it is difficult to apply the H.264 deblocking filter to low-end terminals. Although some technologies to optimize it have been proposed, the complexity is still high for real time implementation. Considering that deblocking filter is applied to all of the vertical and horizontal edges of 4x4 blocks but only some of them actually need to be filtered, in this paper, based on previous encoding or decoding information, we propose a novel deblocking filter algorithm to reduce complexity of exist algorithms by skipping quantities of needless edges. The experimental results show that our proposed algorithm has better performance in subjective video quality and time efficiency compared to JM 10.2 algorithm.

Keywords- H.264/AVC; Deblocking filter; Loop filter

I. INTRODUCTION

H.264/AVC standard provides high compression efficiency compared to previous video coding standards, such as MPEG-4 and H.263, mainly due to various compression algorithms added into this standard [1][2]. For example, block-based discrete cosine transform (DCT) and block-based motion compensated prediction (MCP). For the code efficiency of the H.264, its use has been growing in consumer electronic products for applications such as video telephony and video surveillance.

However, the use of block-based processing, adds visible blocking artifacts to the reconstructed frames. In order to overcome this problem, in-loop deblocking filter is used to remove the blocking artifacts. Although the deblocking filter improves qualities of output video frames, the computational complexity of the H.264 is two or three times higher than that of the H.263 and MPEG-4. In [3], time arranged in H.264 decoder are reported as follows: loop filtering (33%), interpolation (25%), followed by bit-stream parsing and entropy decoding (13%) and reconstruction (13%). As it can be seen among above, loop filtering has the largest computational complexity.

With the purpose of presenting high performance implementation for real time applications, many efficient hardware architectures for deblocking filter in H.264 have been proposed. A deblocking architecture requires extremely low gate count for meeting the mobile application was illustrated [4] and various fast memory accessing techniques have been proposed [5][6] to satisfy the real-time constraints. But none of these solutions considers the algorithmic enhancements for overall speedup, so the complexity still

cannot be reduced significantly because of the flow of algorithm itself.

On the other hand, there are several new deblocking algorithm in the literature that have resulted in better video quality. For instance, a post deblocking filter has been added into the original algorithm [7], which result in better performance, but there is another time and memory time elapsing either. In [8], projection onto convex set (POCS) approach has been proposed. The idea behind POCS is to iteratively impose different sets of constraints to reduce the blocking artifacts. A set of constraints is applied as long as it is closed convex. Chia-Hung Yeh, Kai-Lin Huang and Shiunn-Jang Chern [9] has introduced a color analysis post-processing deblocking filter to reduce the blocking effects which based on the human color vision and color psychology analysis. However, few methods above mentioned their algorithm complexity.

In order to obtain the sufficient gains to meet real-time requirement, two optimization steps are introduced in this paper by fully utilizing encoding or decoding information. And the rest of this paper is organized as follow. The H.264/AVC deblocking filtering algorithm is reviewed in Section 2. Our proposed novel algorithm is described in Section 3. In Section 4, simulation results are presented and compared with the reference model JM 10.2 software. Finally, the paper ends with conclusion in Section 5.

II. THE H.264/AVC DEBLOCKING ALGORITHM

In H.264/AVC standard, deblocking filter treats every decoded luminance 16x16 macroblock as a filtered unit. As the Figure 1 depicting, boundary strength (BS) is calculated from first vertical edges (a-d), and then horizontal edges (e-h). For each edge, data in the form of horizontal or vertical line of pixels (LOP) is processed 16 times respectively (0-15).

The rules to determine the BS for each LOP has been shown in Table I. In this way, in order to process a 16x16 macroblock, 128 separate operations should be required. And each LOP is applied adaptive filter only if (1) is true. In our complex analysis experiment, BS computation through BS detection has taken up 90% overall time during deblocking process. So our optimization work is to utilize previous information which has been done before delocking filter to avoid complex BS operation.

$$BS \neq 0, |p_0 - q_0| < \alpha, |p_1 - p_0| < \beta, |q_1 - q_0| < \beta \quad (1)$$

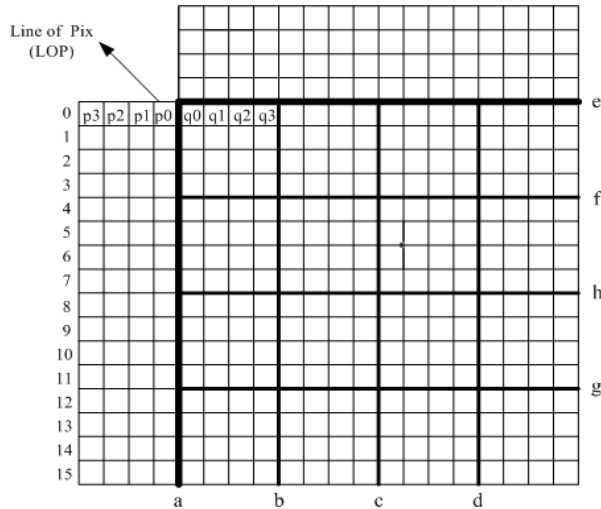


Figure1. The order of computing Bs in a 16x16 macroblock.

TABLE I. DETERMINING BS FOR LOP.

Condition	BS
p or q is intra coded and boundary is a macroblock boundary	4
p or q is intra coded and boundary is not a macroblock boundary	3
neither p or q is intra coded; p or q contain coded coefficients	2
neither p or q is intra coded; neither p or q contain coded coefficients; p and q have different reference frames or a different number of reference frames or different motion vector values	1
otherwise	0

III. OUR PROPOSED OPTIMIZATION ALGORITHM

Since the minimal unit in the previous DCT, intra/inter prediction and MCP are based on 4x4 blocks, one 4x4 block should be a independent unit which include the same coded coefficients, intra/inter mode and motion vector values. According to the Table 1, the four LOP of vertical edge or horizontal edge in one 4x4 block should have the same BS. Thus, Our first optimization step is that we only calculate BS values of LOP 0,4,8 and 12, then copy BS value of LOP 0 to LOP 1,2 and 3; copy BS value of LOP 4 to LOP 5,6 and 7; copy BS value of LOP 8 to LOP 9,10 and 11; copy BS value of LOP 12 to LOP 13,14 and 15 in Figure 1. By doing so, we can decrease nearly 3/4 complexity in BS computation without any video quality loss.

In H.264, there are seven different block size modes (skip, 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4) that are used in the macroblock for inter prediction, as it shown in Figure 2. We select three sequence (foreman, stefan and coastguard) in different quantity parameter (QP) to statistics block mode used in Figure 3.

As depicted in Figure 3, skip mode, 16x16 mode, 16x8 mode, and 8x16 mode have dominated over 60% block size modes in sum. Moreover, the number of skip mode grows swiftly with the increased QP value. We could find that when QP is equal to 40 only the skip mode has already reached approximately 60% of all the block size mode. So these block modes consume the most deblocking time.

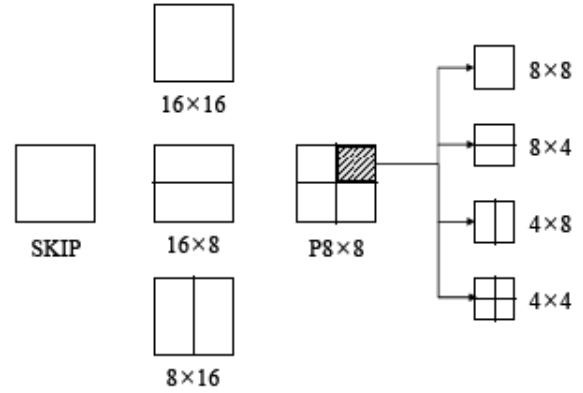


Figure 2. Block size modes used for inter prediction in H.264.

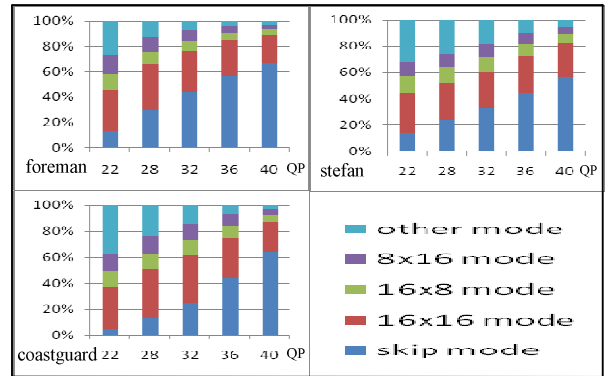


Figure 3. Block size mode used in three sequence.

The H.264 standard divides a frame into a set of large area block size mode (16x16, 16x8, and 8x16) for their internal pixels have changed not as rigidly as other small block size modes. So a separate motion vector is required for every block size mode in Figure 4. According to Table 1, we could draw a conclusion that for 16x16 mode, 16x8 mode and 8x16 mode, their internal pixels have the same motion vector values and are all belong to inter macroblock mode, and hence almost no blocking artifacts boundary will appear in these block size mode interior.

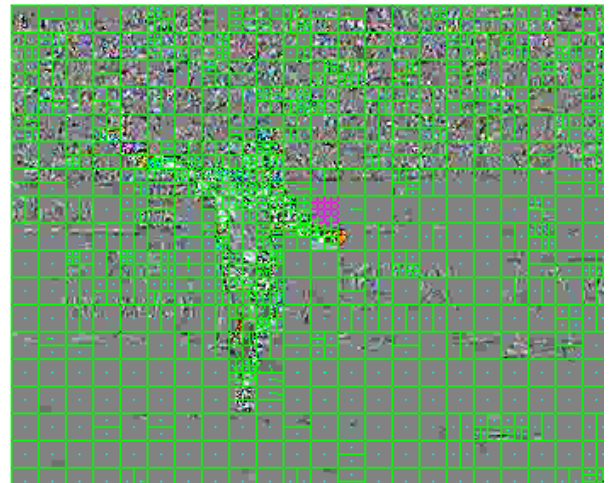


Figure 4. Different block modes and motion vectors assigned in residual of stefan

In the inter prediction, skip mode is very special form. In order to adopt skip mode, macroblock should meet all of the four conditions below:

- block size mode being 16x16 mode;
- reference frame being just one previous;
- motion vector value being zero;
- transform coefficients being all quantized to zero;

Apparently, the conditions above, requested by skip mode have indicated that there no blocking artifacts would generate on the boundary of skip mode. Consequently, if we apply deblocking filter on these numerous skip modes, a amount of time waste will be inevitable.

On the basis of the analysis above, we propose our second optimization approach. Before we apply the deblocking filter on macroblock, we firstly obtain the block size modes information from the previous inter prediction process without any extra computation. If current macroblock is skip mode, we will skip this macroblock directly instead of placing any blocking artifacts detection on it. Due to the large proportion that skip mode macroblocks have occupied, we avoid to devote a lot of needless time during the deblocking filter process. For 16x16 mode, 16x8 mode and 8x16 mode, we only select the boundary where blocking artifacts most frequently take place, to apply deblocking filter. Refer to Figure 1, we only filter vertical edge a and horizontal edge e in 16x16 mode macroblock; filter vertical edge a and horizontal edges e,h in inter 16x8 mode macroblock; filter vertical edges a,c and horizontal edge e in 8x16 mode macroblock. Thus, we avert to filter some internal edges of 16x16 modes, 16x8 mode and 8x16 mode. By doing so we could save plenty of time. So our two steps optimization deblocking algorithm is generally illustrated in Figure 5.

Deblocking filter start

```

If macroblock is skip mode
    break;
Else if macroblock is 16x16 mode
    Deblockedge(a,e);
Else if macroblock is 16x8 mode
    Deblockedge(a,e,h);
Else if macroblock is 8x16 mode
    Deblockedge(a,c,e);
Else
    Deblockedge(a,b,c,d,e,f,g,h);

```

Deblockedge()

```

{
    compute BS for LOP 0 then copy it to LOP 1,2,3;
    compute BS for LOP 4 then copy it to LOP 5,6,7;
    compute BS for LOP 8 then copy it to LOP 9,10,11;
    compute BS for LOP 12 then copy it to LOP 13,14,15;
    apply adaptive filter on LOP;
}

```

Deblocking filter end

Figure 5. Our two steps optimization deblocking algorithm.

IV. EXPERIMENTAL RESULTS

For our simulations, in order to measure the performance of the deblocking filter proposed in this paper, the JM 10.2 software is selected as a reference for our experiment. The test sequence parameters are showed in Table II. We use the commonly used tool PSNRCalculator to measure objective video quality Peak Signal to Noise Ratio (PSNR). As we could see in Table III, the average of PSNR reduction for our proposed algorithm is slightly lost compared to JM 10.2. Inter VTune Performance Analyzer software is used for weighing time reduction in contrast to JM 10.2. And our test platform is Inter 686 processor with 1024MB memory. Since an absolute time for each sequence may vary with the different test platforms, we use time consume percentage between our proposed deblocking algorithm and deblocking algorithm in JM 10.2 to measure time efficiency. In Figure 6, we test fast (stefan) activity sequence filter time when QP=22,28,32,36,40, and our proposed algorithm gains time-saving improvement 43.34%, 48.72%, 57.1%, 71.7%, 68.9% respectively, compared to JM 10.2. The results show that our proposed algorithm has obvious time reduction in all the QP values. Especially, the time used in our proposed algorithm decreases rapidly with the QP values increasing. As the bit rate is inversely proportional to the QP values, our approach has better performance in low bit rate coding.

TABLE II. PARAMETERS FOR TEST SEQUENCE.

Sequence name	foreman, stefan, coastguard
Frame size	CIF (352x288 pixels)
Encoded frames	50
Reference frames	5
Video format	30fps, IPPP sequence
Fast motion estimation	enabled
Fast mode decision	enabled
Motion estimation search range	16

TABLE III. Δ PSNR FOR DIFFERENT TEST SEQUENCE.

QP	Foreman	stefan	coastguard
	Δ PSNR	Δ PSNR	Δ PSNR
22	-0.02dB	-0.02dB	0.02dB
28	-0.01dB	-0.02dB	-0.02dB
32	-0.03dB	0.01dB	-0.02dB
36	-0.06dB	-0.01dB	-0.02dB
40	-0.04dB	0.01dB	0.02dB
average	-0.03dB	-0.006dB	-0.004dB

$$(\Delta \text{ PSNR} = \text{PSNR}(\text{Our proposed algorithm}) - \text{PSNR}(\text{JM}))$$

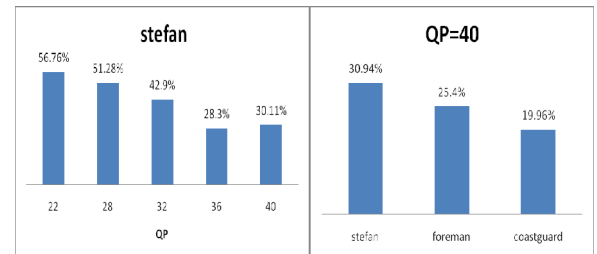


Figure 6. Our proposed deblocking algorithm's processing time percentage compare to JM 10.2 deblocking algorithm's processing time.



Figure 7(a). Original 27th frame.



Figure 7(b). Reconstructed 27th frame QP=40 without deblocking filter.



Figure 7(c). Reconstructed 27th frame QP=40 with JM 10.2 deblocking algorithm.



Figure 7(d). Reconstructed 27th frame QP=40 with our proposed deblocking algorithm.

We also make a control group in low (coastguard) activity sequence, medium (foreman) activity sequence and fast (stefan) activity sequence in figure 6. And we find that our method process faster in foreman and coastguard because low activity sequence has more large area block size modes. Figure 7(a)-(d) are illustrated to measure the sequence subjective video quality visually. In Figure 7(b), we could feel a mass of blocking artifacts there. With JM 10.2 deblocking algorithm in Figure 7(c), blocking artifacts has been removed. However, the whole picture has been blurred in contrast with our proposed algorithm in Figure 7(d).

V. CONCLUSION

In this paper, a novel deblocking filter algorithm was proposed for video real time encoding or decoding in H.264. The first optimization step copies one BS value of LOP to that of the other three LOP in the same 4x4 block in order to avoid repetitive calculations. The second optimization step skips some needless edges so as to not waste time to detect BS values on these edges. According to the results, it was verified that computational cost can be decreased considerably with better subjective video quality, especially, in low rate coding or low activity sequence.

ACKNOWLEDGMENT

This research was supported by the Introduction of Advanced International Forestry Science and Technology Program from the State Forestry Administration, China (Grant No. 2005-4-04) and the College Students Creative Research Program from the Ministry of Education of China (Grant No. GCS08016).

REFERENCES

- [1] ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC, "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification," JVT Doc. JVTG050, 2003.

- [2] T. Wiegand, G. J. Sullivan, G. Bjontegard, A. Luthra, "Overview of the H.264/AVC Video Coding Standard," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560-576, Jul. 2003.
- [3] M. Horowitz, A. Joch, F. Kossentini, etc. "H264/AVC Baseline Profile Decoder Complexity Analysis," *IEEE Transactions on Circuit and System for Video Technology*, Vol. 13, No. 7, pp. 704-716, Jul. 2003.
- [4] Vladimir Ciric, Ivan Milentijevic, "Area-Time Tradeoffs in H.264/AVC Deblocking Filter Design Mobile Device," *9th International Symposium on Signal Processing and Its Applications*, pp. 1-4, Feb. 2007.
- [5] S.-C. Chang, W.-H. Peng, S.-H. Wang, T. Chiang, "A Platform Based Bus-interleaved Architecture for De-blocking Filter in H.264/MPEG-4 AVC," *IEEE Trans. on Consumer Electronics*, vol. 51, no. 1, pp. 249- 255, Feb. 2005.
- [6] B. Sheng, W. Gao, D. Wu, "An implemented architecture of deblocking filter for H.264/AVC," *IEEE International Conference on Image Processing*, vol. 1, pp. 665-668, Oct. 2004.
- [7] Yao-Min Huang, Jin-Jang Leou, Ming-Hui Cheng, "A Post Deblocking Filter for H.264 Video," *Computer Communications and Networks, ICCCN 2007*, pp. 1137 - 1142, Aug. 2007.
- [8] J. Zou, H. Yan, "A Deblocking Method for BDCT Compressed Images Based on Adaptive Projections," *IEEE Transaction on Circuits and Systems for Video Technology*, Vol. 15, No. 3, pp. 430-435, Mar. 2005.
- [9] Chia-Hung Yeh, Kai-Lin Huang, Shiunn-Jang Chern, "Deblocking Filter by Color Psychology Analysis for H.264/AVC Video Coders," *Intelligent Information Hiding and Multimedia Signal Processing*, 2008. pp. 802 - 805, Aug. 2008.