Optimizing the Deblocking Algorithm for H.264 Decoder Implementation

Ken Kin-Hung Lam

Abstract

In the emerging H.264 video coding standard, a deblocking/loop filter is required for improving the visual quality of the decoded frames. These filters attempt to remove the artifacts introduced by the block-based operations, which are discrete cosine transform and motion compensation prediction. Although the deblocking filter performs well in improving the subjective and objective quality of output video frames, they are usually computationally intensive. Among the deblocking algorithms proposed, Adaptive Deblocking Filter is regarded as the least computationally complex one because only addition, shift and comparison operations are involved. In this paper, adaptive deblocking filter was analyzed and found that more than 90% of the computation resources were spent on computing the edge strengths. In this paper, a scheme exploiting the similarity of edge strengths among adjacent normals is proposed and implemented. The simulation results show that quality measured in both of signal-to-noise-ratio(SNR) and universal quality index(UQI) close to the original algorithm can be achieved with almost 50% of total computation reduced.

1 Introduction

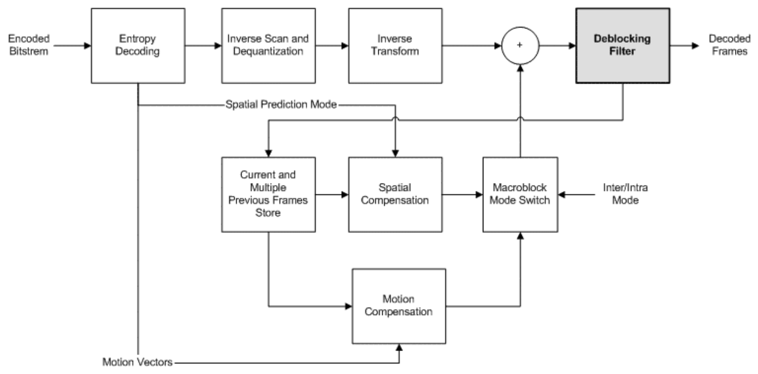


Fig. 1 Block diagram of H.264 decoder

H.264 recommendation is an emerging video coding standard drafted for very low bit rate video communication applications such as video conferencing on mobile phones. Unlike its predecessor, H.263, a deblocking filter process is required for improving the visual quality of the decoded frames [1]. Each video frame is divided into 16x16 pixels blocks called macroblocks. The deblocking filter is applied to all the edges of 4-by-4 pixel blocks in each macroblock except the edges on the boundary of a frame or a slice. For each block, vertical edges are filtered from left to right first, and then horizontal edges are filtered from top to bottom. The decoding process is repeated for all the macroblocks in a frame.

A major challenge in designing blocking artifact detection is the detection of true edges in an image. Blindly applying a lowpass filter would remove most of the block artifacts, but blurs the image as well.

According to an analysis of run-time profiles of decoder sub-functions reported in [2], this deblocking filter process is the most computationally intensive part that consumed as much as one-third of computational resources of the decoder.

The algorithms proposed so far are based on complex mathematical derivations to identify and remove the block artifacts. Although significant improvement in subjective and objective quality can be achieved with them, their computation and implementation complexity is so high that it prohibits them from being adopted directly in a real time application such as H.264 decoder in a cost effective manner.

2 Key Algorithms Proposed

Among the various algorithms proposed, there are 3 key classes of algorithms which based on projection on convex sets (POCS) [4][5], weighted sum of pixels across the block boundaries[6], and adaptively applying different filters[3].

The POCS-based algorithms originate from the image restoration algorithm proposed in [5]. It was proposed the first time in [4] to apply to deblocking of images. Two sets of constraints are imposed on an image to restore it from its corrupted version. After defining the transformations between the constraints, the algorithm starts in an arbitrary point in one of the sets, and projects iteratively among them until convergence. The iterative nature of this class of algorithms prohibits it from applying on a real-time system because the time to convergence is unbounded. Moreover, the projections involve filtering of the picture and transformation to frequency domain, which take unacceptably large amount of computation resources.

In the weighted sum of pixels algorithms, the value of each pixel in the picture is recomputed with a weighted sum of itself and the other pixel values which are symmetrically aligned with respect to block boundaries. In [6], the authors proposed a scheme of including 3 other pixels which are taken from the block above, to the left and the block above the left block. The weights are determined empirically and can either be linear or quadratic. This algorithm is essentially performing a filtering operation on every pixel in a picture. It is expected that a very high performance platform would be required to implement this algorithm in a real-time application.

Among the three key classes of algorithms, adaptive filtering, which will be discussed in detail in Section 3, is regarded as the one with lowest computation complexity because only addition, shift and comparison operations are used.

3 Adaptive Deblocking Filter

The deblocking process can be separated into two stages. In the first stage, the edges are classified into different edge strengths according to the pixel values along the normals to the edges. In the second stage, different filtering schemes are applied according to the strengths obtained in stage one. In [3], the edges are classified into 3 types to which no filter, weak 3-tap filter and strong 5-tap filter are applied. The algorithm is adaptive because the thresholds for edge classification are based on the quantization parameters included in the relevant blocks. An edge will only be filtered if the difference between the pixel values along the normal to the edge, but not across the edge, is smaller than the threshold. For high detail blocks on the side of edges, the differences are usually larger and strong filtering is seldom applied to preserve the details. As the threshold increases with the quantization parameters, the edges across the high detail blocks will be filtered eventually because a high coding error is assumed for large quantization parameters.

Since the edges are classified before processing, strong filtering can be replaced by weak filtering or even skipped. Also the filtering is not applied to every pixel but only to those across the edges. A significant amount of computation can be saved through the classification. In view of its relatively lower computational complexity, this is a good starting point to explore for further reduction in amount of computation.

4 Adaptive Deblocking Filter Analysis

The exploration started with an analysis on the distribution of computation resources spent on each stage. JM 9.3[8], which is an H.264 reference codec, was instrumented to gather statistics on the operations performed by each stage of the deblocking filter. Since H.264 was proposed for low bit rate video communication, the analysis was limited to encoding of video sequences into bit streams of 4kbps, 8kbps, 16kbps and 32kbps. Six QCIF format video sequences, including *Akiyo*, *Carphone*, *Coastguard*, *Foreman*, *Mobile* and *Salesman*, were encoded. QCIF format sequences were chosen because they fit in the screens of most of the video phones nowadays. The analysis results are reported in average number of “operations” spent on each stage as shown in Table 1. As addition, shift and comparison are similar in complexity, they are tallied under a common term “operations”. Though averages are reported, the distribution is still reflected accurately because the percentage of operations spent on edge strength computation only differ by less than 3% among different encoding bit rates.

Table 1 Analysis result on distribution of operations in adaptive deblocking filter

Sequence	**Average no. of operations spent on edge strength computation**	**Average no. of operations spent on edge filtering**	**Percentage of operations spent on edge strength computation**
Akiyo	30806000	763106	97.58
Carphone	30189536	2226984	93.13
Coastguard	29931530	2284007	92.91
Foreman	30459125	2012419	93.80
Mobile	29472854	2826595	91.25
Salesman	30613028	1107708	96.51

The analysis results show that more than 90% of the total computation resources were spent on edge strength computation. Obviously, we should focus on optimizing the edge strength computation algorithm for a significant reduction in computation.

5 The Heuristic

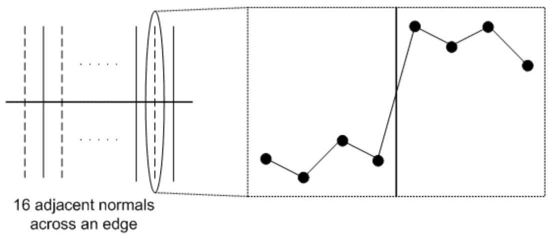


Fig. 2 Proposed edge strength computation scheme

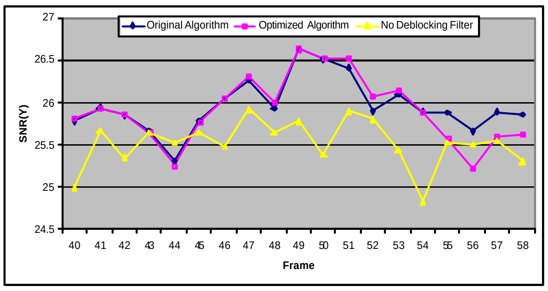
Overall strength of an edge is computed from the edge strengths of 16 normals across the edge. In regions where adjacent pixels are highly correlated, such as smooth surfaces, adjacent *normals* on an edge may result into similar edge strengths. Instead of computing the edge strength for every normal, some of the edge strength computations may be skipped. A possible scheme is illustrated in Fig. 2. While the edge strengths of the normals shown as dash lines are computed, the edge strengths of the normals shown as solid lines are skipped and taken directly from the edge strength of the normals left to them. This new scheme was implemented in JM 9.3, and simulation results obtained are discussed in the following section.

6 Results

Table 2 Simulation result with optimized algorithm

Sequence	**Average no. of operations spent on edge strength computation**	**Average no. of operations spent on edge filtering**	**Percentage of operations reduced**	**Percentage difference of operations spent on edge filtering**
Akiyo	15405598	756268	48.80	-0.8961
Carphone	15096780	2224871	46.57	-0.09483
Coastguard	14961475	2299732	46.42	0.6885
Foreman	15226669	2026466	46.87	0.6980
Mobile	14724358	2859537	45.56	1.1655
Salesman	15309228	1089931	48.30	-1.6049

In Table 2, simulation results with the optimized algorithm are shown. The percentage of operations reduced was calculated by comparing the total sum of operations with the corresponding total sum reported in Table 1. The average number of operations spent on edge filtering between the original algorithm and the optimized algorithm was also compared. A positive percentage means that more operations were performed in the optimized algorithm for edge filtering, and a negative percentage means that less operations was done. While more than 45% of operations were reduced with the optimized algorithm, the difference in number of operations spent on edge filtering is less than 2%. This is desirable because either under-filtering or over-filtering leads to suboptimal results or unnecessary distortions.



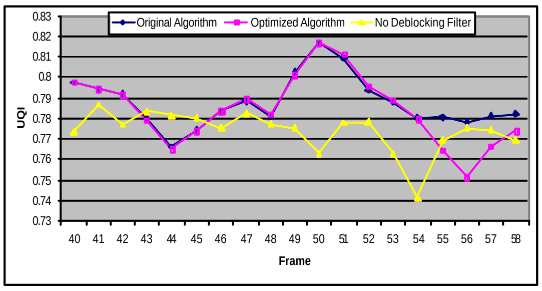
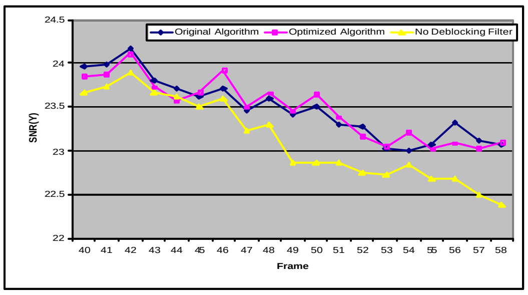


Fig. 3 SNR and UQI achieved in "Foreman" encoded at 8kbps



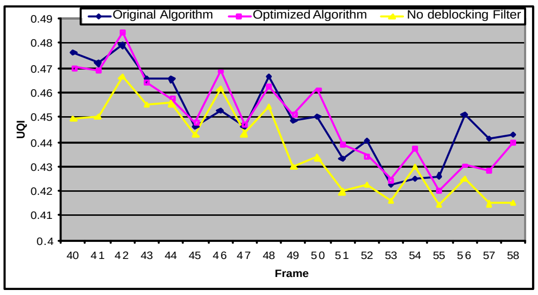


Fig. 4 SNR and UQI achieved in "Coastguard" encoded at 8kbps

The quality of the decoded frames were measured with both signal-to-noise-ratio(SNR) and universal quality index(UQI)[7]. Two examples are shown in Fig. 3 and Fig. 4. The SNR and UQI achieved by the optimized algorithm are shown to be closely approximating those achieved by the original algorithm. In frame 56 of the *Foreman* sequence, the SNR and UQI dropped below the quality achieved without any filtering. It was found that more bits were used to encode the previous frames such as frame 52. With bit rate control, fewer bits were available when frame 56 is coded and so a worse quality resulted.

7 Conclusions

A heuristic exploiting the similarity of edge strengths between adjacent normals was proposed to reduce the computations on overall edge strength. The simulation results with the optimized algorithm show that total number of operations on edge filtering is different from that of the original algorithm by less than 2%. SNR and UQI achieved by the optimized algorithm were also closely approximating those achieved by the original algorithm. Almost 50% of total number of operations was saved without significant impact on the quality of the decoded frame comparing to the original algorithm.

8 References

[1] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC), Joint Video Team (JVT), Doc. JVT-G050, Mar. 2003.

[2] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol.13, no.7, pp. 704-716, Jul. 2003.

[3] P. List, A. Joch, J. Lainema, G. Bj?ntegaard, and M. Karczewicz, "Adaptive deblocking filter," *IEEE Transactions on Circuits and Systems for Video Technology*, vol.13, no.7, pp. 614-619, Jul. 2003.

[4] A. Zakhor, "Iterative procedures for reduction of blocking effects in transform image coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol.2, no.1, pp. 91-95, Mar. 1992.

[5] D. C. Youla and H. Webb, "Image restoration by the method of convex projections: Part 1 - Theory," *IEEE Transactions on Medical Imaging*, vol.1, no.2, pp. 81-94, Oct. 1982.

[6] A. Z. Averbuch, A. Schlar, and D. L. Donoho, "Deblocking of block-transform compressed images using weighted sums of symmetrically aligned pixels," *IEEE Transactions on Image Processing*, vol.14, no.2, pp. 200-212, Feb. 2005.

[7] Z. Wang, and A. C. Bovik, "A universal image quality index," *IEEE Signal Processing Letters*, vol.9, no.3, pp. 81-84, Mar. 2002.

[8] Joint Video Team Reference Software, Version 9.3 (JM9.3), http://iphome.hhi.de/suehring/tml/download/