ADAPTIVE DEBLOCKING FILTER FOR DCT CODED VIDEO

Salim Chebbo**, Philippe Durieux#, Béatrice Pesquet-Popescu *

# NXP Semiconductors, France
* Signal and Image Proc. Department, Telecom-ParisTech, France

ABSTRACT

In this paper we propose a new post-processing algorithm to reduce the blocking artifacts in block based video coding, while maintaining a low computational complexity. This algorithm is mainly based on an adaptive conditional two-dimensional filter, derived from the combination in horizontal and vertical directions of a simple two-mode conditional 1-D filter. By taking into account spatial pixel activities as well as the quantization step, the local degradation of the image is assessed to select the appropriate filtering mode. Experimental results demonstrate that the proposed algorithm improves both subjective and objective picture quality.

1. INTRODUCTION

Block-based coding has become the most popular solution for video compression, and has been nowadays adopted in a large number of video standards such as MPEG-1 [1], MPEG-2 [2], MPEG-4 [3] and H264 [4]. However, the coarse quantization of the frequency transform components, assembled in non-overlapping blocks which are processed independently, generates visible grid patterns in the decoded frames. This grid is well aligned on the transform block boundaries for Intra frames. The motion compensation process spreads this grid into the P and B frames; therefore, blocking artifacts can be located anywhere inside these images.

Numerous deblocking algorithms have been proposed for compressed still pictures, to alleviate the blocking artifacts. In [5] and [6], a two dimensional signal-adaptive filtering was proposed, working well for still coded pictures, where the blocking artifacts are always located at the block boundaries. However, in video coding, blocking artifacts may occur at any position in the frame. Thus these algorithms are not sufficient to remove the video artifacts. Iterative methods based on Projection On Convex Set (POCS), have also been proposed [7],[8], but these methods are not suitable for real time video coding due to their computational complexity. Other post-processing algorithms have been proposed for video coding, most of them applying a one dimensional vertical filter to remove horizontal edges, followed by another one dimensional horizontal filter to remove vertical edges [9], [10], resulting in a large number of redundant operations. Furthermore, these methods operate on previously decoded frames as a processing unit, where each decoded picture should be fetched and stored in external memory twice for applying filters in both directions, which increases drastically the memory bandwidth. In [11], a fast two dimensional deblocking filter is introduced for H.264/AVC video coding; the proposed 2-D algorithm is derived from the normative H.264 1-D filter, and improves the decoding speed by 40% at the cost of losing only 0.15% of the objective video quality.

In this paper we present a new two dimensional deblocking filter for MPEG-1/2/4 and H264. Besides the decoding speed improvement induced with the use of 2-D filters, by decreasing the number of required filtering operations, the processing is applied on a MB basis, achieving a large gain in the memory bandwidth. Moreover, the filtering mode decision is conducted over each 4x4 block boundaries, using a six pixel vectors, which gives more accurate results than the existing methods.

In the remainder of this paper, an overview of the proposed algorithm is described in Section 2. In Section 3, we present the new method based on 2-D filters. In Section 4, several simulation results are examined, and our conclusions drawn in Section 5.

2. OVERVIEW OF THE PROPOSED METHOD

In our approach, we propose a new two dimensional post-processing filter, which exploits the quantization step, extracted during the decoding process, to assess local image degradation and then processes degraded areas depending on the pixel activities.

The proposed algorithm operates on an MB basis. Decoded MBs output by the decoder are filtered on the fly before being stored in memory. In that way, the current
MB is processed, using a single MB local buffer, in which the left MB is available for filtering the left MB border. Concerning the filtering of the top MB border, only the 3 bottom lines of the upper MBs have to be fetched from memory. The overhead, in term of memory bandwidth, induced by this video post-processing algorithm is then drastically reduced compared to filtering algorithms applied on all vertical edges in horizontal direction, and then on all horizontal edges in vertical direction.

To cover most of block-based video codec, the proposed method filters at 4×4 block boundaries. Thus, our algorithm can be applied on MPEG as well as on H264 coded sequences. Moreover, applying a 4×4 block filtering is especially more effective on P or B pictures where original transform block grid has been spatially shifted. Filtering is conducted in a manner that each pixel is filtered only once instantly in vertical and horizontal directions with a 2-D filter. This results in reducing the number of required arithmetical operations (multiplier, adder, and shifter).

Besides the complexity in terms of the number of instructions, the complexity of a filter should also be considered in terms of required memory access events. This remark holds for both software and hardware solutions. On one hand, memory accesses will be translated into LOAD/STORE instructions as well as instructions for address computation. On the other hand, a hardware implementation requires the instantiation of an embedded memory for storing pixels to be filtered, and each memory access will require one cycle to load or store one pixel into that embedded memory. The throughput of a hardware filter implementation is in fact much more likely to be limited due to congestion on the embedded memory rather than by the computational complexity of the filter itself. Taking into account these implementation constraints, it becomes obvious that opting for a 2-D filter drastically reduces the overall algorithm complexity as the blocks to be filtered need to be scanned only once using a 2-D filter, whereas a 1-D filter applied successively in both directions requires two scans of the same block.

3. PROPOSED DEBLOCKING ALGORITHM

For each extended MB of 20x20 pixels, a filtering mode decision matrix is generated before filtering. Filtering modes are computed based on the variation, referred to as activity, of vertical and horizontal six-pixel vectors at each 4×4 block boundary. From these 2 filtering modes is derived the proper 2-D filter to be applied on the desired pixel. Once filtering mode is established, the 2-D filtering process itself is carried out. In order to alleviate the number of computations requested for 2-D filtering, we process the MB in two passes. First pass filters so called anchor pixels around block edges using an adaptive set of 2-D filters with a maximum of 16 taps. In the second pass, the remaining pixels are filtered with 2-D adaptive filters using a maximum of 7 taps, which takes as input the filtered pixels from the first pass.

3.1. Decision modes

In order to select the appropriate filtering mode, an activity factor is assigned to 4 contiguous pixel locations inside each vector of six pixels at the 4×4 block boundaries, as described in Fig.1. This activity is derived from the following formula:

\[ F(P) = \sum_{i=1}^{5} \Psi(p_i - p_{i+1}) \]  

(1)

where

\[ \Psi(\Delta) = \begin{cases} 0, & |\Delta| \leq T_1 \\ 1, & \text{otherwise} \end{cases} \]

\( T_1 \) is a fixed threshold, \( P \) represents the six-pixel vector, and \( p_i \) are the pixel values. Thus the activity factor \( F(P) \) represents the number of detected edges inside the vector \( P \). According to \( F(P) \), the vector \( P \) can be classified into three filtering mode types, as illustrated in Fig.2. In this figure, \( T_2 \) represents a fixed threshold, max and min are the maximum and minimum values of \( P \), and \( G(QP) \) is a functions of the quantization step and the video codec in use.

If the strong filtering mode is chosen, all pixels in the given vector \( P \) are filtered with an appropriate filter, as described later. Otherwise, if the default mode is selected, only the two pixels around the block edge (\( p_7 \) and \( p_8 \)) are filtered. So in that latest case, the filtering mode for \( p_7 \) and \( p_8 \) is reset to no filter, whereas the filtering mode for \( p_3 \) and \( p_4 \) remains set to default.
The filtering mode decision procedure is reiterated over the entire macroblock at each 4x4 block boundary in both vertical and horizontal directions, before applying filters. Thus, for each extended MB we generate a filtering mode matrix with 20x20 values, representing the filtering modes for both directions. These values have the XY format, where X represents the horizontal filtering mode, and Y the vertical filtering mode. X and Y take their values from \{N,D,S\}, where N means no filtering, D means default filter, and finally S means strong filter.

According to the relative pixel position, with reference to the 4x4 vertical and horizontal edges, we distinguish four sets of pixels as shown in Fig.3. Each set is characterized by the filtering modes assigned to its pixels. Pixels located around the horizontal block boundary, belonging to G2, can have only one of the following filtering modes: \{NN,ND,NS,SN,SD,SS\}, as the horizontal default filter is not possible for G2 pixels. Similarly the possible filtering modes of other sets are:

G1: \{NN,NS,ND,DD,DS,SN,SD,SS\}
G3: \{NN,NS,ND,DS,SN,SS\}
G4: \{NN,NS,SN,SS\}

3.2. Filtering

The proposed filtering process specifies a scanning order throughout the MB. For clarity, we introduce the notion of a Filtering Window (FW) to designate a 6x6 pixel box centered at the intersection of four 4x4 pixel blocks as illustrated in Fig.4 (a). This FW is first placed at the upper left corner of the MB, and is then shifted according to the scanning order given in Fig.4 (b). The 8x8 block edges are filtered first, followed by the remaining 4x4 block edges. This special processing order allows the algorithm to better consider blocking artifacts in video coded with 8x8 block DCT, like MPEG-4.

To avoid multi-filtering, only 16 pixels (dark pixels in Fig.4(a)) in each FW are filtered using the proposed 2-D filter. In order to further reduce the complexity of our non-separable filters, we process the MB in two passes. During the first pass, we filter only eight pixels in each FW (grey pixels in Fig.5(a)), therefore by shifting the FW all over the MB according to the previously defined scanning order, we obtain a half-filtered macroblock as shown in Fig.5(b). In this diagram, the dark gray pixels represent formerly filtered pixel from the upper and left MBs, while the light gray pixels are the filtered pixels after running the first pass on current MB. The white regions represent the remaining unfiltered pixels. These pixels are filtered later in a second pass, with a simplified set of filters as defined below.

3.2.1. First filtering pass:

In the first pass, in each FW only the eight pixels \{p_{12}, p_{13}, p_{21}, p_{22}, p_{24}, p_{31}, p_{32}, p_{34}\} belonging to G2 and G3 sets are filtered, as depicted in Fig.5(a). As previously stated, the possible filtering modes for G2 and G3 pixels are: \{NN,ND,NS,SN,SD,SS\} + \{NN,NS,ND,DS,SN,SS\}.
For filtering modes with an N (no filter) in any direction (ND, NS, DN, SN), only one dimensional filters are required. For instance, ND and DN modes apply a 1-D default filter on the target pixel in vertical and horizontal direction respectively. The ND mode can be assigned to the pixels belonging to G2, \( \{p_{21}, p_{23}, p_{24}, p_{25}\} \). In this case, the filter is applied vertically on the target pixel. The updated values of \( p_{12} \) and \( p_{13} \) are computed as:

\[
p_{12} = (p_{13} + 5p_{12} + 3p_{13} - p_{14}) > 3
\]  
\[
p_{13} = (p_{14} + 5p_{13} + 3p_{12} - p_{11}) > 3
\]

\( p'_{12} \) and \( p'_{13} \) are computed in a similar manner. As in the ND filtering mode of the pixels belonging to G3 (\( \{p_{21}, p_{23}, p_{24}, p_{25}\} \)), the filtered pixel values are computed symmetrically to the ND filtering.

If the NS or SN mode is selected, then a strong 1-D filter is applied to the target pixel in one of both directions. In this filtering mode, the updated value of \( p_{ij} \) is computed as follows:

\[
p_{y(i,j \in [1,2,3,4])} = \frac{1}{8} \sum_{k=3}^{k=3} b_{y(i,dir*k+j*(1-dir*k))}
\]

\[
V_{m} = \begin{cases} 
    p_{m1}, m < 0 \\
    p_{m0}, 0 \leq m \leq 5 \\
    p_{m5}, m > 5,
\end{cases} 
\]

\[
b_y : (-3 \leq k \leq 3) \in \{1,1,2,1,1\}, \text{ and } \text{dir} \text{ is equals 1 in horizontal direction, and 0 in vertical direction.}
\]

In the case where the filtering mode belongs to \{DS,SD,SS\}, a 2-D filtering is applied on the desired pixel. The proposed 2-D filters are simplified versions from the combination of the horizontal and vertical 1-D filters described in the previous section. Indeed, in order to preserve a small amount of computations, the weighting matrix of the 2-D filter is simplified, some coefficients having a small weight are cut, and others are rounded, while preserving similar filter characteristics. Fig. 6 and Fig. 7 show the simplified 2-D filters used to process \( p_{21} \) and \( p_{12} \) in SS, DS, and SD modes. Other G2 and G3 pixels are filtered in the same way.

3.2.2. Second filtering pass:

At the end of the first pass, pixels belonging to G2 and G3 are filtered throughout the MB. During this second pass, we filter the remaining pixels, belonging to G1 and G4, by applying the appropriate filter according to the pre-assigned filtering mode, and using the updated pixels from the first pass. In Fig. 8, dark pixels represent pixels filtered during the first pass, while white pixels are updated according to their assigned filtering mode as follows:

\[
p_{y} = (2p_{(i-1,j) + p_{(i+1,j)}} + p_{y}) > 2
\]

\[\text{DN:}\]
\[
p_{y} = (p_{(i-1,j) + 5p_{(i+1,j)}} + 3p_{(i+1,j) - p_{(i+1,j)}}) > 3
\]

\[\text{SD:}\]
\[
p_{y} = (6p_{(i-1,j) + 4p_{(i+1,j)} + 4p_{(i+1,j)}} + 2p_{(i+1,j) - p_{(i+1,j)}}) > 4
\]

\[\text{DD:}\]
\[
p_{y} = (8p_{(i-1,j)} + 4p_{(i+1,j)} + 4p_{(i+1,j)} + p_{y}) > 3
\]

\[\text{SS:}\]
\[
p_{y} = (2p_{(i-1,j) + 2p_{(i+1,j)} + 2p_{y} + p_{(i+1,j)} + p_{(i+1,j)}}) > 3
\]
For symmetric filtering modes, the filtered values of \( p_j \) are simply computed in a symmetric manner.

By using these filters, the number of computations needed per pixel is decreased, in comparison with the 16 tap filters used in the first pass, without losing in picture quality.

### 4. SIMULATION RESULTS

Objective performance is measured with the peak signal to noise ratio (PSNR). Simulations are performed using the MPEG-4 VM coder. Several video test sequences are coded using various QP values in order to generate coded bit-streams at various bitrates. Each sequence consists of 300 frames; only the first frame is coded as an intra frame, while the following ones are coded as P frames.

The PSNR results of three different sequences are listed in Table 1. In Fig.9, a snapshot view from the ‘Foreman’ sequence in CIF format coded at 229Kbs is given. The selected frame is a P one, where removing blocking artifacts is more challenging due to the fact that artifacts are located at uncertain positions, and not at the original 8x8 grid. Obviously, the proposed filter overcomes the MPEG-4 MV one, mainly thanks to the adaptive filtering of propagated artifact, which MPEG-4 VM does not take into account. It also better preserves fine details by performing local activity estimation on smaller areas.

The proposed algorithm has been designed to operate with widely used block-based video codecs, including H.264, which clearly outperforms older video codecs in terms of objective as well as subjective picture quality for a given bit rate. However at very low bit rate, some annoying artifacts can remain despite the normative in-loop filter. Applying the proposed method on such sequences eliminates these remaining artifacts, and improves the picture quality both subjectively and objectively. Fig. 10 shows the results after applying the proposed filter to the CIF Foreman sequence encoded at 73.92 Kbs (15 HZ), with a fixed QP equal to 37.

<table>
<thead>
<tr>
<th>Sequences (CIF/ 30Hz)</th>
<th>Bit-rates (kbs)</th>
<th>QP</th>
<th>VM</th>
<th>VM+ filter</th>
<th>VM+ proposed filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreman</td>
<td>229</td>
<td>29</td>
<td>29.16</td>
<td>29.15</td>
<td>29.21</td>
</tr>
<tr>
<td>Foreman</td>
<td>260</td>
<td>20</td>
<td>30.24</td>
<td>30.29</td>
<td>30.34</td>
</tr>
<tr>
<td>Coastguard</td>
<td>237</td>
<td>29</td>
<td>26.80</td>
<td>26.84</td>
<td>26.88</td>
</tr>
<tr>
<td>Coastguard</td>
<td>316</td>
<td>20</td>
<td>28.18</td>
<td>28.25</td>
<td>28.28</td>
</tr>
<tr>
<td>News</td>
<td>94</td>
<td>29</td>
<td>29.24</td>
<td>29.40</td>
<td>29.47</td>
</tr>
<tr>
<td>News</td>
<td>115</td>
<td>20</td>
<td>30.01</td>
<td>31.17</td>
<td>31.20</td>
</tr>
</tbody>
</table>

Table 1: PSNR comparison of the test sequences for the MPEG-4 VM, the MPEG deblurring filter, and the proposed filter.

![Deblocking result for Foreman sequence](image_url)
Fig.10: Deblocking results for Foreman sequence. (a) Original frame. (b) H.264 decoded frame (c) Deblocked with the proposed filter.

The algorithm complexity is estimated via the number of executed instructions, which is reduced by more than 40% for ADD/SUB, and 25% for SHIFT instructions, with respect to the MPEG-4 VM deblocking filter.

5. CONCLUSION

This paper presented a new two dimensional non-separable deblocking filtering procedure, targeting MPEG-1/2, MPEG-4 and also H.264 encoded streams. Both objective and subjective quality tests illustrate the efficiency of the proposed algorithm, on various sequences encoded at different bit rates with MPEG-4 and H.264 codecs. Finally, while providing consistent picture quality improvement, the low complexity of this algorithm makes it suitable for real time post processing and hardware implementation.

6. REFERENCES


