A Joint Architecture of Error-Concealed Deblocking Filter for H.264/AVC Video Transmission

Wen-Ping Lee, Tsu-Ming Liu and Chen-Yi Lee
Department of Electronics Engineering, National Chiao Tung University
1001, Ta Hsueh Road, Hsinchu 300, Taiwan, ROC
E-mail: mb@si2lab.org, mingle@si2lab.org, cylee@si2lab.org

ABSTRACT

In this paper, we propose new spatial error concealment (SEC) method, error-concealed de-blocking filter (ECDF), for a real-time decoding system over an error-prone or wireless channel. There are several advantages of ECDF. The first is that ECDF can conceal I frames without the need of flexible macroblock ordering (FMO). Second, the hardware cost of interpolation can be saved. Third, the required information for error concealment (EC) only includes the pixels in the top and left neighbors of current corrupted macroblock (MB). Hence, we can conceal the corrupted MB without requiring the information in the right and bottom side, leading to the reduction of memory space as well as bandwidth. The implementation results show that the hardware cost of ECDF can be saved about 30% compared to direct implementation. Without the FMO, the proposal gains 1.3dB in PSNR compared to bilinear interpolation in JM9.8.

1. INTRODUCTION

Of all modalities desirable for future mobile multimedia systems, high-quality motion video over a reliable transmission is the most demanding. However, the transmitted visual quality may suffer abruptly because the channel deteriorates due to fading, co-channel interference, and signal attenuations. To deal with the transmission errors over an error-prone channel, much effort has been invested to improve the error-robustness in the source decoding procedures, such as error resilient tools and ECs.

So far, many researches focus on EC for the corrupted video bit-stream over an error-prone or wireless channel. The EC can be divided into two groups: SEC and temporal error concealment (TEC). The SEC exploits the neighboring correct pixels to recover the corrupted MB with the property of high correlation between the closing pixels. This kind of EC is used for corrupted MBs in I frames or I MBs in P frames. Unlike the SEC, TEC tries to estimate the lost motion vectors (MV) in corrupted MBs. The basic TEC replaces the missing MV with zero MV, the MVs of an adjacent correctly received MB, average or median of all such available adjacent MVs. The performance of TEC could be improved if we use matching measure to determine the lost MV instead of replacement. Instead of using the average or median over all neighbor motion vectors, the side-matching algorithm [2] uses the motion vector which belongs to one of neighbor MB’s motion vector to make the block boundaries smooth. The winning prediction MV is the one which minimizes the side match distortion $d_{sm}$. As shown in Eqn. (1), $d_{sm}$ is the sum of absolute Y pixel value differences of the IN-block and OUT-block pixels in Figure 1. However, TEC is not always adequate for concealing errors in video sequences. This is especially true for video sequences with irregular motion, abrupt scene changes, and intra-coded image frames. To the viewer, poor spatial concealment of error regions leads to the error propagation in the subsequent frames.

The algorithms widely used in SEC are bilinear interpolation (BI) and directional interpolation (DI) [3]. As shown in Figure 2(b), BI replaces each pixel with the weighting average of nearest pixels in the neighboring MBs with four directions. The weights used here are the inverse distance of the missing and raw pixel. DI consists of edge detection, edge direction ranking and 1-D interpolation. In the first step, the edge detection uses Sobel mask to determine the gradient and edge direction in the neighborhood of a corrupted MB. With the information, edge direction ranking would determine the possible edge through the corrupted MB. Finally, the 1-D interpolation along the specified direction is applied to conceal the lost MB with weights that is inversely proportional to the distance of missing and raw pixel as shown in Figure 2(a). BI is suitable for texture due to the property of smooth, and DI has better performance if there are real edges exist in the corrupted MB. Hence, an algorithm which combines the DI and BI with mode decision has been presented [5]. It conceals MB with BI to avoid the creating false edges or with DI to preserve the real edges according to the directional entropy of neighboring edges.

*Work supported by the National Science Council of Taiwan, R.O.C., under Grant NSC 94-2215-E-009-046-
For the spatial error concealment, most algorithms, such as BI or DI, belong to frame-based EC due to the thing that usage of pixels in four direction pixels. In order to keep the neighboring macroblocks available, the encoder has to support the FMO. Besides, the decoding system needs additional memory or local buffer to store the pixels on the boundary of neighboring MBs. The BI or DI which are widely used in I frames could have an acceptable performance with FMO, but the performance would decrease a lots without FMO. Here we propose a new SEC which only needs the pixels on the top and left side stored in the slice memory used by de-blocking filter [4]. The proposed SEC is combined with de-blocking filter to reduce the hardware cost and reuse the memory capacity.

The remainder of this paper is structured as follows. The proposed SEC method is described in Section 2. The proposed method consists of edge detection, replacement and smoothing. The simulation and implementation results are given in Section 3. Finally, conclusions have been made in Section 4.

2. ERROR-CONCEALMENT DEBLOCKING FILTER

For SEC, it is significant to preserve the existing edges without creating new strong ones. For the case, the algorithm of DI is given to prolong the edges entering to the corrupted MB. But for the texture area, the performance of BI is better than DI. So the algorithm which uses DI and BI with mode decision has been presented to improve the visual quality according to the edge activities [5]. However, such kind of algorithm needs more memory bandwidth or local buffers to keep the neighboring pixels. In order to reduce the memory and hardware cost, we propose an ECDF which combines the de-blocking filter in H.264/AVC with EC.

As we know, the de-blocking filter doesn’t work when the decoded MB is erroneous, and the EC will not be activated when the decoded MB is correct. Based on the design concepts, we combine the de-blocking filter and EC to reduce hardware cost. Moreover we limit the pixels used to conceal corrupted MBs to the top and left neighboring 4x4-blocks. As shown in Figure 3, the pixels used in EC are the dotted 4x4-blocks stored in slice memory [4].

The ECDF consists of edge detection, replacement, smoothing and de-blocking filter which is based on our previous work. The ECDF filters all edges in a correct or corrupted MB. In Figure 4, the dotted blocks are additional parts for error concealment capabilities and the signal, Corrupted MB, is transmitted from error detection. ECDF does de-blocking or EC according to the signal. The pixel buffer stores the pixels from content memory or slice memory [4]. 1-D de-blocking filter and boundary strength are the algorithms standardized in H.264. The pixels Q on the right (bottom) side are replaced pixels in a specified direction determined by edge detection. Therefore, the edges extended into the corrupted MB can be preserved slightly. Further, we modify the algorithm of boundary strength and filterSamplesFlag to reconstruct texture area well.

In general, Sobel mask is used to determine the magnitude of gradient as shown in Eqn. (2) and edge slope of existing edge in the neighboring 4x4-blocks.

\[ G_x = (S_x)^T F, \quad G_y = (S_y)^T F \]

\[ |G| = \sqrt{G_x^2 + G_y^2}, \quad \text{slope} = \frac{G_y}{G_x} \]  

As shown in Figure 5, the Sobel mask moves along the arrowhead in the neighboring blocks for each erroneous 4x4-block. The dotted grid is the location of center in Sobel mask. Taking Figure 5(a) as an example, we operate the Sobel mask to calculate the edge information 15 times because of 15 grids on the path Sobel mask move along. Then we use the information to predict the corrupted 4x4-block. The gradients are used to judge if there is a real edge or not. If the gradient is larger than the fixed threshold evaluated by simulations, the edge is regarded as real one. And the edge slopes determine which replacing mode to be used. Therefore, we can support multiple edges extended from the top and left neighboring blocks of this corrupted 4x4-block.


2.2 Replacement

When the proper edge is determined in the corrupted block, we use main direction replacement instead of interpolation to reduce the hardware cost. By ranking the edge slope with thresholds evaluated from simulations, four replacing modes can be used to recover the missed pixels as Figure 6 shows.

<table>
<thead>
<tr>
<th>Vertical mode</th>
<th>Horizontal mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>E F G H J K L M</td>
<td>E F G H J K L M</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>e</td>
<td>f</td>
</tr>
<tr>
<td>g</td>
<td>h</td>
</tr>
</tbody>
</table>

One of the modifications is that we force the signal, filterSamplesFlag, to be true when the decoded MB is erroneous. With the strong filter (BS equals 4) specified in H.264, 6 pixels can be smoothed per edge. For the weak filter (BS<4), there are only 4 pixels can be filtered. The other modification is that we force the boundary strength (BS) which is specified in H.264 standard, to be true when the decoded MB is corrupted.

\[
\text{filterSamplesFlag} = (\text{BS} > 0 \& \& |p_i - q_i| < \alpha \& \& |p_j - p_i| < \beta \& \& |h_k - q_j| < \beta)
\]

2.3 Smoothing

After replacing the missing pixels on the right (bottom) side in the corrupted 4x4-block, the de-blocking filter starts to smooth the edges of the corrupted block to improve the visual quality. There are two modifications in the algorithm of de-blocking filter in H.264. One of the modifications is that we force the boundary strength (BS) equals 4 when the decoded MB is erroneous. With the strong filter (BS equals 4) specified in H.264, 6 pixels can be smoothed per edge. For the weak filter (BS<4), there are only 4 pixels can be filtered. The other modification is that we force the signal, filterSamplesFlag, which is specified in H.264 standard, to be true when the decoded MB is corrupted.

There is one problem for edge detection if the neighboring blocks of corrupted 4x4-block are concealed ones such as Figure 5(b). The predicted edge in corrupted block may be not proper due to the edge information in the neighboring blocks strongly depends on replacing mode. It means that one erroneous block recovered using horizontal replacing mode has the horizontal edge information. Taking Figure 7 as an example, the neighboring blocks (B, C) are replaced ones. If we conceal each block by normal order, the edge information in blocks (B, C) strongly depend on replacing mode due to that blocks (B, C) are replaced by other neighboring pixels. On the other hand, the blocks (B, C) can be smoothed by filtering edges (2, 3, 5, 6) with hybrid order before we use the edge information in blocks B, C to predict the proper edge in corrupted block (dotted block).

3. PERFORMANCE EVALUATIONS

3.1 Simulation Results

For simulations, we consider the CIF test pattern which is encoded with JM9.8 using dispersed FMO and slice size of 198 MBs. With dispersed FMO, there are different type of corrupted MB we can conceal to see the total performance in directional or texture area. Further, we assume that error detection is perfect when decoding process and EC is operated in MB-based processing. It means that there is no information used on the right (bottom) side of one corrupted MB in EC. From the simulation results below, the PSNR of proposed algorithm is about 24dB and the one of BI in JM9.8 is 22.70dB.

![Figure 8: PSNR of the proposed algorithm and BI in JM9.8.](image)

<table>
<thead>
<tr>
<th>Technique</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>With hybrid order</td>
<td>24.04dB</td>
</tr>
<tr>
<td>With smoothing</td>
<td>23.9dB</td>
</tr>
<tr>
<td>With edge detection and replacing</td>
<td>23.7dB</td>
</tr>
<tr>
<td>BI without FMO</td>
<td>22.70dB</td>
</tr>
</tbody>
</table>

![Figure 9: The improvement of PSNR.](image)
3.1.1 Fix some problems or modify thresholds in edge detection

The first method of improving the performance is to modify the edge detection to determine the edges extended from neighborhood more correctly. The thresholds of the edge direction and gradient have to be modified. From the Figure 10(c), we can see that there are still some edges predicted not well.

3.1.2 Add additional replacing mode for other edge

The second method of improving the performance is using much replacing modes with detail edge direction. For the edge with about 20 degree in the Figure 10(b), we use horizontal replacing mode to recover. Here results some reduction of performance.

3.1.3 Get the balance between smoothing blocks and preserving real edges

Third, we can see that the real edges in Figure 10(a) are still smoothed by ECDF. Here we should have some thresholds which are based on the gradients of neighboring blocks to determine whether the corrupted 4x4-block should be smoothed or not. By doing so, we can preserve the real edges extended from neighborhood or smooth the corrupted block adaptively.

The original processing time per correct MB of de-blocking filter is 250 cycles.

On the other hand, the gate count of ECDF becomes 22.74K due to the hardware cost of edge detection. Here we neglect the hardware cost of two parts, replacing and smoothing, because that the two parts are composed of control signals almost. And the processing time is still 250 cycles which equals to the one of de-blocking filter [4] for a correct or corrupted MB. Therefore, the ECDF achieves 67.6% of hardware cost as compared with the direct implementation. And the power consumption can be reduced due to the reduction of hardware cost and external memory access.

<table>
<thead>
<tr>
<th>Items</th>
<th>De-blocking filter + EC</th>
<th>ECDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gate count</td>
<td>33.64K(0.18um)</td>
<td>22.74K(0.18um)</td>
</tr>
<tr>
<td>Working frequency</td>
<td>100Mhz</td>
<td>100Mhz</td>
</tr>
<tr>
<td>Required pixels from</td>
<td>1-pixel on the right</td>
<td>none</td>
</tr>
<tr>
<td>external memory</td>
<td>side near the corrupted MB</td>
<td></td>
</tr>
</tbody>
</table>

Figure 10: (a) The real edges also smoothed by de-blocking filter. (b) The edge with about 20 degree is replaced by horizontal mode. (c) Mistakes on edge detection.

3.2 Implementation Results

For the portable real-time device, the power consumption and cost are more significant than other issues. In order to make a comparison about these topics, we implement some major parts in the algorithms of BI and ECDF.

With the 0.18um process, the gate count of the BI is about 7.8K with clock period equals 10ns. Besides, the implementation of BI needs additional hardware cost for local buffer and memory bandwidth to load the neighboring pixels from the external memory to local buffers. To store all the correct neighboring pixels around the corrupted MB, we need 128-bits (i.e. 16x4+2x8x4) registers. Hence, the total gate count for the de-blocking filter and error concealment becomes about 33.64K (19.64K for the de-blocking filter [4] and 6.2K for the local buffer for BI). And the processing time per corrupted MB becomes 384 cycles due to the pixel-based processing.

Table 1: The comparison of architecture and processing time per MB for the error concealment in H.264/AVC.

5. CONCLUSIONS

In this paper, the ECDF which combines the EC with de-blocking filter in H.264 is proposed. The proposal is easily implemented and integrated into the de-blocking filter in H.264. The hardware cost and memory access can be reduced by sharing the memory used by de-blocking filter based on our previous work and using the de-blocking to support the capability of error concealment instead of interpolation. Although the influence of FMO is significant to EC, not all the decoder supports FMO due to the high complexity. Hence, we focus on the simulations without FMO and try to upgrade the performance of ECDF without FMO to approach the performance of BI with FMO. The PSNR of ECDF is 1.3dB better than BI in JM9.8 without FMO and comparable to BI with FMO. Finally, the proposal reduces 30% of hardware cost compared to direct implementation.

6. REFERENCES