MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Implementation of Edge Map Guided Fuzzy Filtering for Artifact Reduction in Highly Compressed Video

Yao Nie, Hao-Song Kong, Anthony Vetro, Toshiaki Shimada, Noriyuki Minegishi

TR2005-007 December 2005

Abstract

This paper presents an implementation of a post-filtering algorithm for the reduction of vido coding artifacts in highly compressed video based on a fuzzy filtering approach. The implementation considers two fast methods for obtaining the filter weights, which are incorporated into a practical scheme for calculating the filter output

ICCE 2004

8.3-2

IMPLEMENTATION OF EDGE MAP GUIDED FUZZY FILTERING FOR ARTIFACT REDUCTION IN HIGHLY COMPRESSED VIDEO

Yao Nie[1], Hao-song Kong[1], Anthony Vetro[1], Toshiaki Shimada[2], Noriyuki Minegishi[2]

[1] Mitsubishi Electric Research Labs, Cambridge, USA
[2] Mitsubishi Electric Corporation, Kanagawa, Japan

Abstract — *This paper presents an implementation of a post-filtering algorithm for the reduction of video coding artifacts in highly compressed video based on a fuzzy filtering approach. The implementation considers two fast methods for obtaining the filter weights, which are incorporated into a practical scheme for calculating the filter output.*

INTRODUCTION

High compression video coding is employed today in many consumer electronics devices, such as HDD/DVD recorders and digital camera, to provide efficient memory utilization. At high compression ratios, blocking and ringing artifacts become prominent and should be reduced during playback. Many algorithms for artifact reduction have been proposed, e.g., [1]-[3]. However, most methods either introduce undesirable blurring to the images or cannot remove all types of artifacts successfully. To address the problem, an edge map guided fuzzy filtering algorithm [4] has been developed by the authors. In contrast to some existing schemes such as [2], our algorithm doesn't rely on any coding information and achieves superior performance. In this algorithm, edge blocks are first identified and a fuzzy filter is applied to the edge blocks for de-ringing. The output of this filter is defined as:

$$y = \frac{\sum_{j=1}^{N} x_j w_j}{\sum_{j=1}^{N} w_j} = \frac{\sum_{j=1}^{N} x_j \cdot \exp[-(x_c - x_j)^2 / 2\xi^2]}{\sum_{j=1}^{N} \exp[-(x_c - x_j)^2 / 2\xi^2]}, \qquad (1)$$

where N is the window size, x_j are the inputs, and w_j are the filter weights, $x_c \in \{x_1, x_2, ...x_N\}$ is the center pixel in the window, ξ is referred to as the spread parameter. Direct implementation of the filter according to (1) requires evaluating exponential functions, floating-point arithmetic and division operations. Such computation is undesirable for a hardware implementation. Therefore, we propose a scheme to reduce the computational complexity of this approach by avoiding the online evaluation of the exponential function to obtain the filter weights, and ultimately use division-free and fixed-point operations to compute the filter output.

PROPOSED SCHEME

The proposed scheme is composed of three main steps as shown in Figure 1. The input is the set of pixel values in the filtering window centered at the pixel x_c. We assume that each pixel value is represented by an M-bit integer. The scheme computes the fuzzy filter output, y_c, by performing only integer summation, multiplication and bit shift operations, such that:

$$y_c = \frac{\sum_{j=1}^{N} x_j w_j}{\sum_{j=1}^{N} w_j} \approx \left(\alpha \sum_{j=1}^{N} x_j \hat{w}_j\right) \gg U, \qquad (2)$$

where $\hat{w}_1, \hat{w}_2, ..., \hat{w}_N$, α and U are all positive integers, and $\gg U$ means right shifting by U bits. In Step 1, input values are used to obtain the integer-valued filter weights, $\hat{w}_1, \hat{w}_2, ..., \hat{w}_N$. In Step 2, the sum $\sum_{j=1}^{N} \hat{w}_j$ is mapped to an integer normalization factor, α. Finally, in Step 3, the output is computed through integer summation, multiplication and shifting. These steps are described further below.

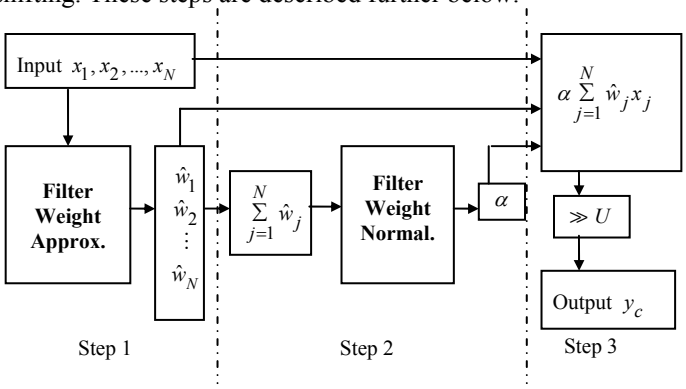

Figure 1. Overview of the proposed scheme.

Step 1: Filter Weight Approximation

Two methods to obtain the integer-valued filter weights are explored: a *Look-Up Table* (LUT) method and a *Linear Approximation* (LA) method. The LUT method requires minimum computation, while the LA method significantly reduces the memory usage. The diagram for the LUT method is show in Figure 2(a), where the input is the pixel intensity pair (x_j, x_c), and the output is the weight, $\hat{w}_j$. The lookup table, $LUT^{\hat{w}}$, is pre-calculated and stored such that $LUT^{\hat{w}}[i] = \left[\exp(-i^2/2\xi^2) \cdot 2^p\right]$, and $i=0,1,...2^M-1$, where $[]$ is the rounding operation and the scaling factor 2^p is chosen to achieve appropriate precision. Thus, in this method, only the pixel value differences need to be computed to obtain the filter weights, but a lookup table of size 2^M needs to be stored.

In LA method, the exponential function is approximated

by a piecewise linear function, $\mu_L(x)$, which is defined as:

$$\mu_L(x) = \begin{cases} 2^p, & 0 \le x \le (2-e^{0.5})\cdot\xi \\ -\left[2^p e^{-0.5}\xi^{-1}\right]x + \left[2^{p+1}e^{-0.5}\right], & (2-e^{0.5})\cdot\xi < x < 2\xi \\ 0, & x \ge 2\xi \end{cases} \quad (3)$$

This piecewise function approximates the scaled exponential function with spread parameter ξ. For fixed spread parameter, ξ, and scaling factor, 2^p, eqn. (3) becomes a linear function with constant integer coefficients and is rather simple to compute. The diagram for the method is shown in Figure 2(b), where $e^{-0.5} \approx 0.6065, 2 - e^{0.5} \approx 0.3513$.

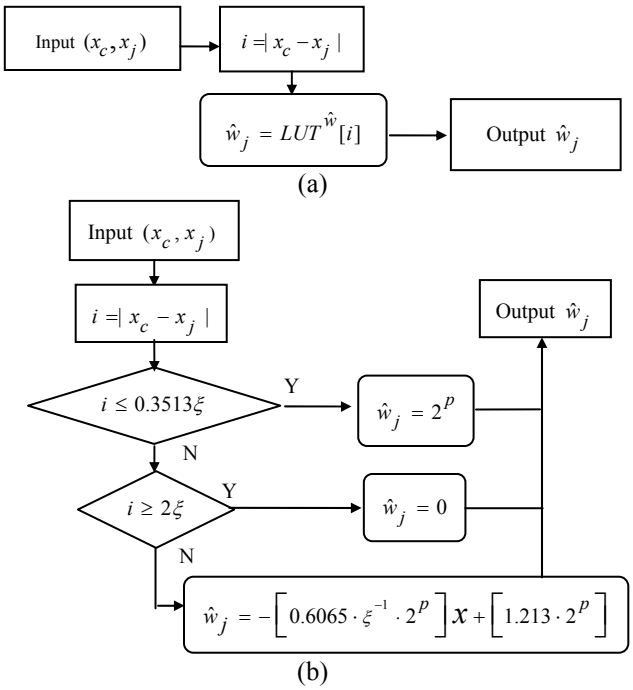

Figure 2. Two methods of approximating the fuzzy filter weights (a) Look-Up Table; (b) Linear Approximation.

Step 2: Filter Weight Normalization

The sum of the fuzzy filter weights is mapped to a normalization factor by using a lookup table, LUT^{α}. Since the size of LUT^{α} is restrained to favor hardware implementation, the sum may need to be rescaled to fit the range of LUT^{α}. The diagram is shown in Figure 3. Here, $\left(\Sigma_{j=1}^{N}\hat{w}_j\right)^*$ is the rescaled sum of the filter weights, whose range is $[2^q, 2^q \cdot N]$, note that $q \le p$. The i^{th} lookup table entry is given by $LUT^{\alpha}[i] = \left[2^u/(i+2^q)\right] = \left[2^u/\left(\Sigma_{j=1}^{N}\hat{w}_j\right)^*\right]$, where $i=0,\ldots,2^q(N-1)$, $u = \log_2 N + q + r$ is chosen so that α has precision $1/2^r$.

Step 3: Filter Output Calculation

Based on the integer weights and the normalization factor determined in the previous steps, an integer-weighted sum is calculated and scaled which is then right-shifted by $U=u+p-q$ bits to yield the final filter output.

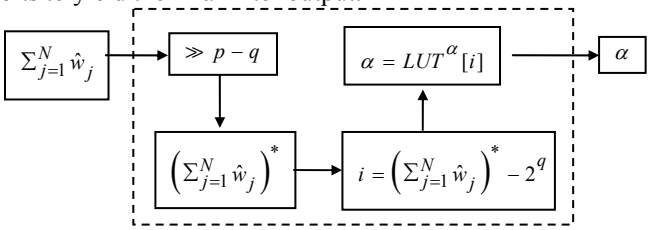

Figure 3. Block diagram of filter weight normalization.

EXPERIMENTAL RESULTS

Two sequences, Mobile Calendar (720x576) and News (352x288), are used to evaluate the proposed approaches. The total number of frames in each sequence is 100 and 150, respectively. Both sequences are compressed using an MPEG-2 TM5 encoder with high quantization. For the fuzzy filtering, a 5x5 window and $\xi = 20$ are used. The division-free and fixed-point implementation (both LUT and LA methods) is compared to the reference scheme, which is a direct implementation of eqn (1). The parameters for the proposed scheme are *p=5, q=4, r=6.* The average PSNR per frame and the total processing time of our software implementation are presented in Table 1. Under these conditions, we observe significantly reduced processing time and no degradation in visual quality.

Table 1. Comparison of PSNR (dB) and Processing Time (sec)

	PSNR (dB)			Time (sec)		
	Ref	LUT	LA	Ref	LUT	LA
Mobile	25.26	25.20	25.34	149.97	46.84	55.66
News	30.61	30.55	30.65	31.31	13.03	14.63

CONCLUSION

This paper presented a feasible solution for the implementation of a fuzzy-filter based post-filtering algorithm for artifact reduction. Two methods are presented to obtain integer-valued filter weights, which are integrated into an overall division-free and fixed-point implementation. The simulation results show that both methods provide significant reduction in complexity and make the algorithm suitable for hardware implementation without any loss of quality. Further information on parameter settings and actual impact on hardware design will be reported in the full-length version of the paper.

REFERENCES

[1] C. B. Wu, B. D. Liu, and J. F. Yang, "Adaptive postprocessors with DCT-based block classifications," *IEEE Trans. Circuits Syst. Video Technol.*, Vol. 13, No. 5, pp. 365-375, May. 2003.

[2] ISO/IEC 14496-2:2001, Information Technology – Generic Coding of Audio-Visual Objects – Part 2: Visual," *2nd Ed., Appendix F3 – Post-processing for coding noise reduction*.

[3] T. Chen, H. R. Wu and B. Qiu, "Adaptive post-filtering of transform coefficients for the reduction of blocking artifacts," *IEEE Trans. Circuits Syst. Video Technol.*, Vol. 11, No. 5, pp. 594-602, May 2001.

[4] H. Kong, Y. Nie, A. Vetro, H. Sun and K. E. Barner, "Coding artifacts reduction using edge map guided adaptive and fuzzy filtering," *IEEE ICME*, Taipei, Taiwan, June 2004.