Adaptive Pre-interpolation Filter for Motion-Compensated Prediction

Jie Dong and King Ngi Ngan
Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong SAR

***Abstract*— [1] The proposed interpolation filter comprises two concatenating filters, adaptive pre-interpolation filter (APIF) and the normative interpolation filter in H.264/AVC. The former is applied only to the integer pixels in the reference frames; the latter generates all the sub-position samples, supported by the output of APIF. The convolution of APIF and the standard filter minimizes the motion prediction error on a frame basis. APIF preserves the merits of the adaptive interpolation filter (AIF) and the adaptive loop filter (ALF) in the key technical area (KTA) software and overcomes their drawbacks. The experimental results show that APIF has comparable or even better performance compared with the joint use of AIF and ALF.**

I. Introduction

As more and more video materials with increased quality and spatio-temporal resolution will be captured and distributed in the near future, the bit-rate produced by the current coding technology, e.g., H.264/AVC, will go up faster than the increased capacity of the wireless and wired network infrastructure [1]. Therefore, a new generation of video coding technology aiming at sufficiently higher compression capability is required. In the past four years, the international standardization bodies, VCEG and MPEG, have been seeking promising techniques regarding the possibility of a major gain in performance to justify the step from H.264/AVC to a new standard. To better evaluate the techniques and retain progress, key technical area (KTA) was developed as the software platform, which used JM11 as the baseline and continuously integrated promising techniques. One of the features making KTA significantly outperform H.264/AVC is adaptive filtering. The techniques related to adaptive filtering can be classified into two categories, adaptive interpolation filter (AIF) and adaptive loop filter (ALF), according to their functions.

AIF improves the interpolation in H.264/AVC. When the target block an MV points to is out of the sampling grid, where the intensity is unknown, the intensities of the positions in between the integer pixels, called sub-positions, must be interpolated. In H.264/AVC, the interpolation filter is fixed. AIF considers the time-varying statistics of video sources, and optimizes the filter coefficients at the frame level such that for each frame the energy of the motion-compensated prediction (MCP) is minimized. All the AIF techniques in KTA use the same linear minimum mean squared error (LMMSE) estimator to obtain the coefficients, but have different support regions and different types of symmetries. Vatis *et al.* [2] develop a 2-D non-separable interpolation filter, in which each sub-position is interpolated by filtering the surrounding 6×6 integer pixels. The filter is in circular symmetry, because the spatial statistical properties are assumed to be isotropic. The directional AIF (D-AIF) [3] restricts the support region to 1-D aligning integer pixels, and therefore is much simpler than [2]. To improve the performance of D-AIF, the enhanced AIF (E-AIF) [4] is proposed, which adds a 5×5 filter for integer pixels and a filter offset to each integer and sub-position pixel. E-AIF is axisymmetric, as the horizontal and vertical statistical properties are thought different.

ALF is placed in the MCP loop after the deblocking process, and is used to restore the degraded frame (caused by compression) such that the MSE between the reconstructed and source frames is minimized. The frame after the ALF process will be stored in picture memory, if it is a reference frame. Like AIF, ALF is calculated and transmitted on a frame basis and the LMMSE estimator is also used. For each degraded frame, ALF can be applied to the entire frame [5] or to local areas. In the latter case, additional side information indicating which areas are to be filtered is transmitted, which can be block-based [6] or quadtree-based [7].

AIF and ALF have their own benefits and limitations, and are mutually complementary in three aspects. First, ALF, applied to integer pixels only, has much lower complexity than AIF, applied to 15 sub-positions[2]. Second, AIF, directly minimizing the energy of the MCP error, significantly reduces the bits used to code the MCP error. On the contrary, ALF, designed to minimize the reconstruction error, cannot benefit the MCP so much as AIF can, although improving the reference capability to some extent. Third, an optimal AIF comprises a set of 15 filters applied to 15 types of sub-positions; each filter comprises real-valued coefficients. In practice, the overhead introduced by the optimal AIF is too large to be transmitted, and therefore approximations have to be made [2], [3], [4], including reducing the support region, imposing the symmetry constraints, and coarsely quantizing the filter coefficients. In [8], we have pointed out that making trade-off between the accuracy of coefficients and the size of side information is the major obstacle to improving the performance of the AIF techniques that code the filter coefficients individually, no matter what kind of trade-off is made. ALF overcomes this drawback of AIF, because only one filter defined for the integer

[1]This work was partially supported by a grant from the Chinese University of Hong Kong under the Focused Investment Scheme (Project 1903003).

[2]To be consistent with the interpolation in H.264/AVC, all the studies in this paper assume 1/4-pixel MCP and each sub-position is supported by the surrounding 6×6 integer pixels.

pixels is transmitted. No trade-off has to be made, which means the coefficients can be quantized in enough precision, while the overhead is still affordable.

According to the above rationales and our tests, the sets of video sources benefiting from AIF and ALF respectively are not exactly the same. To benefit a wider spectrum of video sources and achieve higher coding gain, AIF and ALF can be jointly used. However, the problem is that the complexities of AIF and ALF are additive, whereas the performance improvements are not. Only 1% further bit-rate reduction on average is observed by additionally using either AIF or ALF. Therefore, the joint use of AIF and ALF has technical redundancy.

In this paper, adaptive pre-interpolation filter (APIF) is proposed to preserve the merits of AIF and ALF and at the same time overcome their drawbacks. Pre-interpolation filtering means the integer pixels in the reference frame are filtered, before used to support interpolation. APIF directly minimizes the energy of the MCP error on a frame basis and analytically calculates the coefficients by using LMMSE estimator. As only one filter for integer pixels is used, no trade-off has to be made, which means the filter coefficients can be quantized in high precision without increasing the overhead too much. The experimental results show that APIF has comparable or even better performance compared with the joint use of AIF and ALF. In other words, the joint use of AIF and ALF can be replaced by APIF without any performance loss, and consequently the technical redundancy is removed.

II. Adaptive Pre-interpolation Filter

A. Optimal AIF

As shown in Fig. 1(a), interpolation by definition comprises two steps: upsampling the original reference frame to 16 times the spatial resolution by zero-insertion, which produces undesired spectra in the frequency domain, and removing the undesired spectra by a lowpass filter. The optimal lowpass filter, denoted as h_{opt}, is obtained by the LMMSE estimator, i.e., the energy of the MCP error energy σ_e^2 in (1) is minimized,

$$\sigma_e^2 = \mathscr{E}\left[\left(\sum_{i,j} h(i,j) P_{16}(4x-i+d_x, 4y-j+d_y) - S(x,y)\right)^2\right] \quad (1)$$

where P_{16} is upsampled from the reference frame by a factor 16 using zero-insertion, S is the current frame to be coded, and d_x and d_y are the two components of MV. Letting $\partial\sigma_e^2/\partial h(m,n)$ equal to 0, one can easily obtain h_{opt} and then derive the minimum σ_e^2, since the solution converges to the Wiener-Hopf equations as in (2),

$$\sum_{i,j} h_{opt}(i,j) R_{pp}(i-m, j-n) = R_{ps}(m,n) \quad (2)$$

where R_{pp} and R_{ps} represent the autocorrelation of P_{16} and the motion-compensated cross-correlation of P_{16} and S, respectively. If no symmetry constraint and quantization are imposed, h_{opt} will have 23×23 different real-valued coefficients to be transmitted every frame, which is unaffordable. As introduced in Section I, AIF techniques in practice are the approximations of h_{opt}, which represents different trade-offs between the similarity with h_{opt} and the overhead costs.

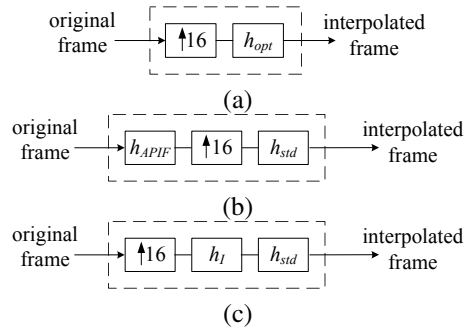

Fig. 1. Diagram of (a) the optimal AIF, (b) APIF, and (c) the upsampled APIF

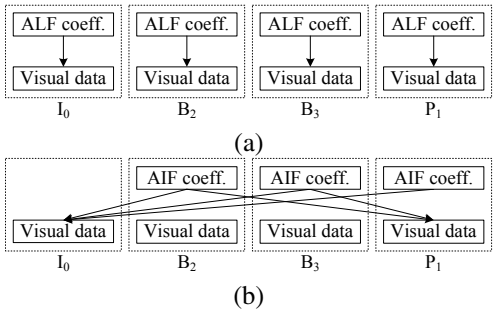

Fig. 2. Use of filter coefficients in (a) ALF and (b) AIF/APIF

B. Concept of the Adaptive Pre-interpolation Filter

The interpolation filter proposed in this paper is composed of two concatenating filters, adaptive pre-interpolation filter (APIF) and the normative interpolation filter in H.264/AVC, as shown in Fig. 1(b). The former is applied to the integer pixels in the reference frame and is optimized on a frame basis; the latter generates all the sub-position samples, supported by the output of APIF. Fig. 1(b) is equivalent to Fig. 1(c), where the relationship between h_{APIF} and h_I is shown in (3), i.e., h_I is the upsampled version of h_{APIF} with zero-insertion.

$$h_I(u,v) = \begin{cases} h_{APIF}(u/4, v/4), & \text{if } u,v \text{ are multiples of 4} \\ 0, & \text{Otherwise} \end{cases} \quad (3)$$

The proposed interpolation filter, denoted as $\tilde{h}$, is the 2-D convolution of h_I and h_{std}, as in (4).

$$\tilde{h}(i,j) = \sum_{u,v} h_I(u,v) h_{std}(i-u, j-v) \quad (4)$$

Ideally, h_I should be designed such that $\tilde{h}$ is exactly the same as h_{opt}. However, such a deconvolution problem is ill-conditioned, which has no solution. We optimize h_I such that $\tilde{h}$ approximates h_{opt}. The details will be presented in Section II-C. The rationales of APIF and ALF are quite different, although both techniques apply filtering only to integer pixels. As shown in Fig. 2(a), the ALF coefficients transmitted in the frame header are used to restore the associated frame. Although the restored frame has improved reference capability for future use, ALF does not directly minimize the energy of MCP error. However, APIF coefficients in the frame header are used for the reference frames, just like AIF coefficients (see Fig. 2(b)). By doing this, the MCP error of the associated frame can be minimized. Different from AIF, which generates sub-position samples directly, APIF coefficients are used

$$E_P = \sum_{i,j}\sum_{m,n}\left[\sum_{u,v} h_I(u,v)h_{std}(i-u,j-v)\right]\left[\sum_{k,l} h_I(k,l)h_{std}(m-k,n-l)\right]R_{pp}(i-m,j-n) - 2\sum_{i,j}\left[\sum_{u,v} h_I(u,v)h_{std}(i-u,j-v)\right]R_{ps}(i,j) \quad (6)$$

$$\sum_{k,l} h_I(k,l)\left[\sum_{i,j}\sum_{m,n} h_{std}(i-a,j-b)h_{std}(m-k,n-l)R_{pp}(i-m,j-n)\right] = \sum_{i,j} h_{std}(i-a,j-b)R_{ps}(i,j) \quad (7)$$

jointly with h_{std}, and interpolation is done by $h_I \otimes h_{std}$. In short, APIF is optimized for minimum MCP error and has less coefficients. The latter means coefficients can be quantized in enough precision without increasing the overhead much.

C. Calculation and Coding of APIF Coefficients

This sub-section introduces how to find the optimal coefficients of h_I, such that interpolation filter $\tilde{h}$ as in (4) achieves the minimum MCP error. In P-frames, the energy of the MCP error in (1) is re-written as in (5).

$$\sigma_e^2 = \mathscr{E}\left[\left(\sum_{i,j}\tilde{h}(i,j)P_{16}(4x-i+d_x,4y-j+d_y)-S(x,y)\right)^2\right] \quad (5)$$

One can substitute (4) for $\tilde{h}$ in (5) and get the energy function E_P (6) for minimization. Letting $\partial E_P/\partial h_I(a,b)$ equal to zero, one will find that the desired h_I is the solution of the equations in (7). The form of the solution in (7) is similar to (2), except that the autocorrelation R_{pp} and cross-correlation R_{ps} in (2) are replaced by their weighted summations, where the weights are the coefficients in h_{std}.

For the bi-directional MCP in B-frames, the energy of the MCP error is re-written in (8), as shown at the bottom of the page, where $P_{16,f}$ and $(d_{x,f}, d_{y,f})$ are the upsampled reference frame and the MV for forward MCP, respectively, and $P_{16,b}$ and $(d_{x,b}, d_{y,b})$ are for the backward case. Similarly, substituting (4) for $\tilde{h}$ in (8), one obtains the energy function E_B in (9) for minimization. In (9), R_{ff} and R_{bb} represent the autocorrelations of the forward and backward upsampled reference frames, $P_{16,f}$ and $P_{16,b}$, respectively; R_{fs} and R_{bs} are the motion-compensated cross-correlations of $P_{16,f}$ and S, and $P_{16,b}$ and S, respectively; R_{fb} and R_{bf} are the motion-compensated cross-correlations of the forward and backward upsampled reference frames. Letting $\partial E_B/\partial h_I(a,b)$ equal to zero, one will finally find that the desired h_I is the solution of the equations in (10), as shown on the top of the next page.

The proposed h_{APIF} has 7×7 taps, and 25 coefficients are coded, based on the symmetry proposed in [6]. APIF coefficients, highly correlated in successive frames, are temporally predicted. Then, the prediction error is uniformly quantized to 2^{12} steps and coded using order-4 Exp-Golomb codes.

TABLE I
TEST CONDITIONS

Test sequence	1280×720 progressive
Sequence structure	IPPP... and IBBP...
Intra frame period	Only the first frame
Entropy coding	CABAC
FME	on
R-D optimization	on
Adaptive rounding	off
QP	I(22, 27, 32, 37) P(23, 28, 33, 38) B(24, 29, 34, 39)
Reference frame	4
Search range	±64
Frame number	58 for IBBP; 60 for IPPP

III. EXPERIMENTAL RESULTS

The proposed APIF is integrated into the VCEG's reference software KTA2.6 and is compared to 2-D non-separable AIF [2], frame-based ALF [6], and the joint use of them, which are subsequently referred to as AIF, ALF, and AIF+ALF, respectively. Table I gives the test conditions; Tables II and III show the coding gain compared with H.264/AVC High Profile, measured by the bit-rate reduction at the same PSNR [9]. As can be seen, AIF outperforms ALF in coding *Crew* and *Jets*, whereas ALF outperforms AIF in coding *City* and *Harbour*. As for APIF, it outperforms the better of AIF and ALF in coding most of the sequences, and provides 1.3%-3.2% more bit-rate reduction on average, compared with either AIF or ALF. Since AIF and ALF benefit different sets of video sequences, the joint use of them can benefit a wider spectrum of video sequences and achieve higher coding gain. However, the problem is that the complexities of AIF and ALF are additive, whereas the performance improvements are about 1% bit-rate reduction, which means the joint use of AIF and ALF has technical redundancy. APIF has comparable or even better performance compared with the joint use of AIF and ALF. In other words, the joint use of AIF and ALF can be replaced by APIF without any performance loss, and consequently the technical redundancy is removed.

On the encoder side, the complexity of implementing APIF mainly lies in the two-pass encoding strategy, just like implementing AIF techniques, such as 2-D non-separable AIF. Other factors, e.g., the number of equations to solve for

$$\sigma_e^2 = \mathscr{E}\left[\left(\frac{1}{2}\sum_{i,j}\tilde{h}(i,j)\left(P_{16,f}(4x-i+d_{x,f},4y-j+d_{y,f})+P_{16,b}(4x-i+d_{x,b},4y-j+d_{y,b})\right)-S(x,y)\right)^2\right] \quad (8)$$

$$E_B = \sum_{i,j}\sum_{m,n}\left[\sum_{u,v} h_I(u,v)h_{std}(i-u,j-v)\right]\left[\sum_{k,l} h_I(k,l)h_{std}(m-k,n-l)\right] \left[\frac{1}{4}R_{ff}(i-m,j-n)+\frac{1}{4}R_{bb}(i-m,j-n)+\frac{1}{4}R_{fb}(i-m,j-n)+\frac{1}{4}R_{bf}(i-m,j-n)\right] - \sum_{i,j}\left[\sum_{u,v} h_I(u,v)h_{std}(i-u,j-v)\right][R_{fs}(i,j)+R_{bs}(i,j)] \quad (9)$$

$$\frac{1}{2}\sum_{k,l} h_I(k,l)\left[\sum_{i,j}\sum_{m,n} h_{std}(i-a,j-b)h_{std}(m-k,n-l)(R_{ff}(i-m,j-n)+R_{bb}(i-m,j-n)\right.$$
$$\left. + \quad R_{fb}(i-m,j-n)+R_{bf}(i-m,j-n))\right] = \sum_{i,j} h_{std}(i-a,j-b)\left[R_{fs}(i,j)+R_{bs}(i,j)\right] \tag{10}$$

TABLE IV

ARITHMETIC OPERATIONS FOR INTERPOLATING ONE FRAME

	Std. Filter in H.264		AIF		ALF		AIF+ALF		APIF		E-AIF	
	multiply	add	multiply	add	multiply	add	multiply	add	multiply	add	multiply	add
Full-pixel	0	0	0	0	49X/K	48X/K	49X/K	48X/K	49X	48X	25X	25X
Half-pixel I	6X	5X	6X	5X	6X	5X	6X	5X	6X	5X	6X	6X
Half-pixel II	6X	5X	36X	35X	6X	5X	36X	35X	6X	5X	12X	12X
Quarter-pixel I	0	X	6X	5X	0 X	X	6X	5X	0 X	X	6X	6X
Quarter-pixel II	0	X	36X	35X	0 X	X	36X	35X	0 X	X	12X	12X
Total	18X	27X	360X	345X	(49/K+18)X	(48/K+27)X	(49/K+360)X	(48/K+345)X	67X	75X	169X	169X

TABLE II

BIT-RATE REDUCTION IN IPPP-CODED SEQUENCES (%)

	AIF	ALF	MIN(AIF,ALF)	AIF+ALF	APIF
Bigships	-10.31	-6.62	-10.31	-9.85	-9.89
City	-14.67	-17.16	-17.16	-18.66	-20.84
Crew	-22.84	-14.54	-22.84	-23.63	-22.07
Harbour	-11.36	-12.44	-12.44	-13.79	-12.34
Jets	-12.73	-7.96	-12.73	-12.71	-13.77
Optis	-6.47	-5.00	-6.47	-7.05	-7.24
Raven	-19.04	-18.41	-19.04	-20.24	-23.67
Sailormen	-10.09	-10.48	-10.48	-12.68	-11.53
Sheriff	-8.97	-8.47	-8.97	-9.72	-10.68
ShuttleStart	-14.71	-10.65	-14.71	-14.41	-17.95
Average	**-13.12**	**-11.17**	**-13.52**	**-14.27**	**-15.00**

TABLE III

BIT-RATE REDUCTION IN IBBP-CODED SEQUENCES (%)

	AIF	ALF	MIN(AIF,ALF)	AIF+ALF	APIF
Bigships	-5.31	-5.98	-5.98	-7.29	-7.85
City	-9.96	-14.46	-14.46	-15.51	-17.63
Crew	-21.36	-9.75	-21.36	-22.88	-19.86
Harbour	-10.16	-13.54	-13.54	-15.10	-13.19
Jets	-6.34	-4.79	-6.34	-6.86	-9.15
Optis	-5.25	-5.65	-5.65	-6.84	-6.49
Raven	-13.26	-13.20	-13.26	-15.78	-17.84
Sailormen	-5.30	-7.90	-7.90	-9.26	-8.05
Sheriff	-5.39	-7.12	-7.12	-7.79	-8.00
ShuttleStart	-7.57	-8.40	-8.40	-8.89	-10.65
Average	**-8.99**	**-9.08**	**-10.40**	**-11.62**	**-11.87**

LMMSE estimator, also influence the encoder complexity, but are relatively trivial. For a decoder, the coefficients received from the bitstream are used for filtering directly, so only the operations used for interpolation are considered. We assume a straightforward implementation. For example, using a 6×6 filter to generate one pixel needs 36 multiplications and 35 additions (shifting is neglected). First, the pixels to be interpolated are classified into five categories, as shown in Fig. 3, according to the order the pixels are generated. Second, we calculate the required numbers of multiplications and additions for interpolating each category of pixels in an entire frame, as shown in Table IV, where X is the number of full-pixels in a frame (equal to the pixel number of any other category) and K is the number of reference frames. Third, the total number of operations used to interpolate a frame to 16 times the size (see the bottom row of Table IV) can be calculated by (6), where N_{Int}, N_{HalfI}, N_{HalfII}, N_{QuarI}, and N_{QuarII} are the operation numbers (multiplication or addition) for the five

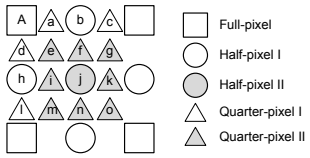

Fig. 3. Five categories of pixels to be filtered or interpolated

categories of pixels, respectively.

$$N_{Total} = N_{Int} + 2N_{HalfI} + N_{HalfII} + 4N_{QuarI} + 8N_{QuarII} \tag{6}$$

Clearly, APIF has much lower complexity than AIF and AIF+ALF, but is of more complex than ALF. However, when the number of reference frame K reduces to one, the complexities of APIF and ALF become the same.

IV. CONCLUSION

The paper proposes the adaptive pre-interpolation filter (APIF), which is applied only to integer pixels in the reference frames and achieves the minimum MCP error on a frame basis when concatenated with the normative interpolation filter in H.264/AVC. The experimental results show that APIF can replace the joint use of AIF and ALF without any performance loss, thus removing the technical redundancy.

REFERENCES

[1] "Vision and requirements for high-performance video coding (HVC)," ISO/IEC JTC1/SC29/WG11 document N10361, 2009.

[2] Y. Vatis and J. Ostermann, "Adaptive interpolation filter for H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, pp. 179–192, Feb. 2009.

[3] D. Rusanovskyy, K. Ugur, A. Hallapuro, J. Lainema, and M. Gabbouj, "Video coding with low-complexity directional adaptive interpolation filters," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, pp. 1239–1243, Aug. 2009.

[4] "Improved filter selection for B-slices in E-AIF," ITU-T Q.6/SG16 (VCEG) document VCEG-AI38, Jul. 2008.

[5] "Adaptive (wiener) filter for video compression," ITU-T Q.6/SG16 (VCEG) document COM16-C437, Apr. 2008.

[6] "Block-based adaptive loop filter," ITU-T Q.6/SG16 (VCEG) document VCEG-AI18, Jul. 2008.

[7] "Quadtree-based adaptive loop filter," ITU-T Q.6/SG16 (VCEG) document COM16-C181, Jan. 2009.

[8] J. Dong and K. N. Ngan, "Parametric interpolation filter for motion compensated prediction," *IEEE Int'l Conf. Image Process. '09 (ICIP09)*, Cairo, Egypt, Nov. 2009.

[9] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," in ITU-T SG16/Q6, VCEG-M33, Apr. 2001.