J. Vis. Commun. Image R. 22 (2011) 697–703

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

Adaptive pre-interpolation filter for high efficiency video coding ☆

Jie Dong *, King Ngi Ngan

Department of Electronic Engineering, The Chinese University of Hong Kong, Shatin, New Territories Hong Kong Special Administrative Region, China

A R T I C L E I N F O

Article history:
Available online 4 January 2011

Keywords:
Pre-interpolation
Adaptive interpolation filter
Adaptive loop filter
H.264
KTA
HEVC

A B S T R A C T

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 at the same time overcomes their drawbacks. The experimental results show that APIF outperforms either AIF or ALF. Compared with the joint use of AIF and ALF, APIF provides comparable performance, but has much lower complexity.

1. 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 January 2010, the standardization bodies, VCEG and MPEG, jointly issued a formal call for proposals (CfP) [2] for the new standard, tentatively named high efficiency video coding (HEVC). Actually, the preparatory work began four years ago. Since 2005, VCEG and MPEG have been seeking promising techniques with a major gain in performance to advance from H.264/AVC to a new standard. To better evaluate the techniques and stimulate progress, key technical area (KTA) [3] was developed as the software platform, which used H.264/AVC's test model JM11 [4] as the baseline and continuously integrated promising techniques. One of the features making KTA significantly outperform H.264/AVC is adaptive filtering. The related techniques 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 and Ostermann [5] 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) [6] restricts the support region to 1-D aligned integer pixels, and therefore is much simpler than [5]. To improve the performance of D-AIF, the enhanced AIF (E-AIF) [7] 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 to be different. Apart from reducing the support region and imposing symmetry constraints, all coefficients in the existing AIF techniques are quantized to 512 levels. However, 9 bits are not enough to represent AIF coefficients. Nevertheless, the required bits for coding these coefficients are still significant especially at low bit-rates.

ALF is placed in the MCP loop after the deblock filtering, and is used to restore the degraded frame (caused by compression) such that the MSE between the reconstructed and source frames is minimized. A reference frame after the ALF process will be stored for future use. Like AIF, ALF is calculated and transmitted for each frame and the LMMSE estimator is used. For each degraded frame, ALF can be applied to the entire frame [8,9] or to local areas. The former is known as frame-based ALF. In the latter case, additional side information indicating which areas are to be filtered is transmitted, which can be block-based [10] or quadtree-based [11].

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

* Corresponding author. Tel.: +852 3163 4340.
E-mail address: jdong@ee.cuhk.edu.hk (J. Dong).

1047-3203/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jvcir.2010.12.008

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.[1] 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 it improves 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 [5–7], including reducing the support region, imposing the symmetry constraints, and coarsely quantizing the filter coefficients. In [12], we 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 integer 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 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 1.5% 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.

To further improve the coding efficiency of AIF and ALF and better fulfill the requirements of HEVC, this paper proposes an interpolation filter comprising two concatenating filters, adaptive pre-interpolation filter (APIF) and the 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 MCP error on a frame basis. APIF's coefficients are analytically calculated using the LMMSE estimator, as for the AIF and ALF's coefficients. APIF preserves the merits of AIF and ALF in the KTA software, including lower complexity than AIF, optimization for minimum MCP error, and less coefficients. The experimental results show that APIF outperforms either AIF or ALF. Compared with the joint use of AIF and ALF, APIF provides comparable performance, but has much lower complexity.

The remainder of the paper is organized as follows. APIF is proposed in Section 2. Section 3 shows the experimental results, followed by the conclusion in Section 4.

2. Adaptive pre-interpolation filter

2.1. 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 inserting zero-valued samples, which produces undesired spectra in the frequency domain, and removing the undesired spectra by a lowpass filter. In AIF, the optimal lowpass filter, denoted as h_{opt}, is obtained by the LMMSE estimator, which means the energy of the MCP error energy σ_e^2 in (1) is minimized,

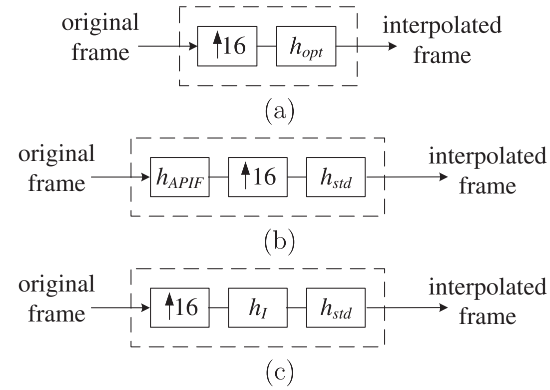


Fig. 1. Diagram of (a) the optimal AIF, (b) APIF, and (c) the upsampled APIF.

$$\sigma_e^2 = \mathcal{E}\left[\left(\sum_{i=-11}^{11}\sum_{j=-11}^{11} h(i,j)P_{16}(4x-i+d_x,4y-j+d_y)-S(x,y)\right)^2\right] \quad (1)$$

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. The MCP error energy is calculated with all the MVs for the current frame known; therefore, motion estimation is performed before starting coding the current frame. The range of h's indices i and j from -11 to 11 is determined by h's size, i.e., 23×23. The reason that h has 23×23 size has been explained in Section III.C of [13].

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),

$$\sum_{i=-11}^{11}\sum_{j=-11}^{11} h_{opt}(i,j)R_{pp}(i-m,j-n) = R_{ps}(m,n) \quad (2)$$

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} in (2) will have 23×23 different real-valued coefficients to be transmitted every frame, which is unaffordable. AIF techniques in practice [5–7] are approximations of h_{opt}, which represent different trade-offs between the similarity with h_{opt} and the overhead costs, as introduced in Section 1. However, making such trade-offs is the major obstacle to improving the performance of the AIF techniques that code the filter coefficients individually [12].

2.2. Concept of the adaptive pre-interpolation filter

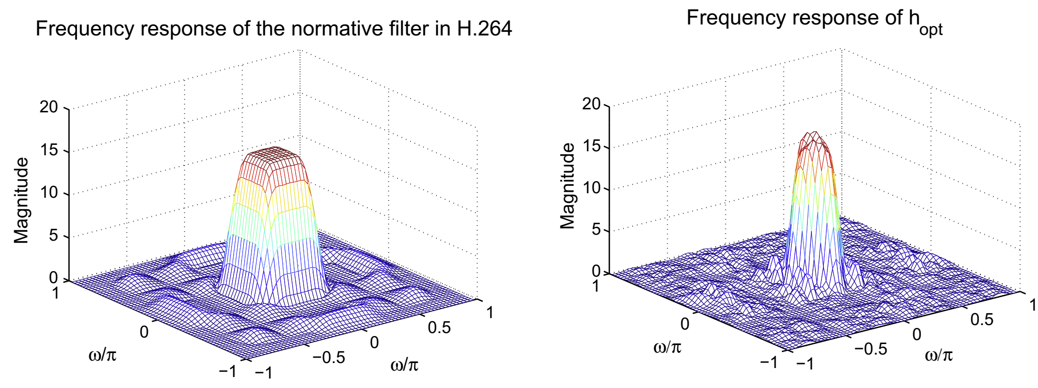
The interpolation filter proposed in this paper is composed of two concatenating filters, adaptive pre-interpolation filter (APIF) and the normative interpolation filter in H.264/AVC, as shown in Fig. 1(b). The former is applied to the integer pixels in the reference frame and is optimized on a frame 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).

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

[1] 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.



(a) (b)

Fig. 2. Frequency responses of (a) the interpolation filter in H.264/AVC and (b) a typical h_{opt}.

Ideally, h_I should be designed such that $\tilde{h}$ is exactly the same as h_{opt}. To interpret this ideal case in the frequency domain, the frequency response of h_{opt}, denoted as H_{opt}, should be the product of the frequency responses of h_I and h_{std}, denoted as H_I and H_{std}, respectively. However, one cannot obtain H_I by dividing H_{opt} by H_{std}; consequently, one cannot obtain the optimal h_I by applying inverse Fourier transform to an impractical H_I. There are two reasons. First, H_{std}, as shown in Fig. 2(a), has zeros, which means H_I, obtained by dividing H_{opt} by H_{std}, has poles and thus does not represent an FIR system. Second, even if H_{std} does not have zeros, i.e., H_I does not have poles, h_I, the inverse Fourier transform of H_I, usually has infinite size and cannot be used in practice. Therefore, rather than making $\tilde{h}$ the same as h_{opt}, which is a ill-conditioned deconvolution problem and has no solution, we optimize h_I such that $\tilde{h}$ approximates h_{opt}. The details will be presented in Section 2.3. Noticing that H_{std} (see Fig. 2(a)) is almost ideal with cut-off frequency $\pi/4$, we can predict that the frequency response of the optimized h_I within $[-\pi/4, \pi/4]$ will well approximate H_{opt} (see Fig. 2(b)).

Both APIF and ALF apply filtering only to the integer pixels of a frame. If the frame is a reference frame and needed to be interpolated for MCP, interpolation filters by APIF and ALF are equivalent to $h_I \otimes h_{std}$ and $h_{ALF} \otimes h_{std}$, respectively. However, the rationales of APIF and ALF are quite different. As shown in Fig. 3(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. On the contrary, APIF coefficients in the frame header are used for the reference frames, just like AIF coefficients (see Fig. 3(b)). By doing this, the MCP error of the APIF's associated frame can be minimized. At the same time, different from AIF coefficients, which are used to generate sub-position samples directly,

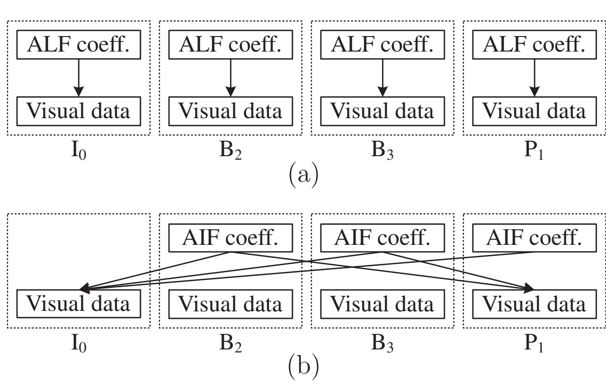


Fig. 3. Use of filter coefficients in (a) ALF and (b) AIF/APIF.

APIF coefficients are used jointly with h_{std}, and interpolation is done by $h_I \otimes h_{std}$. In short, APIF has three advantages: lower complexity than AIF, optimization for minimum MCP error, and less coefficients. The advantage of less coefficients means that one does not need to trade off the coefficients' precision for a smaller frame header, and thus improves the rate-distortion (R-D) performance [12].

2.3. Calculation and coding of APIF coefficients

This section introduces how to find the optimal coefficients of h_I, such that the 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).

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

One can substitute (4) for $\tilde{h}$ in (5) and get the energy function E_P in (6) for minimization.

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

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).

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

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),

$$\begin{aligned} \sigma_e^2 = \mathcal{E}\Bigg[\Bigg(&\frac{1}{2}\sum_{i,j}\tilde{h}(i,j)(P_{16,f}(4x-i+d_{x,f}, 4y-j+d_{y,f}) \\ &+ P_{16,b}(4x-i+d_{x,b}, 4y-j+d_{y,b})) - S(x,y)\Bigg)^2\Bigg] \end{aligned} \tag{8}$$

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.

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

Table 1
Coefficient's symmetry in the proposed APIF.

0	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	23	22	21
20	19	18	17	16	15	14
13	12	11	10	9	8	7
6	5	4	3	2	1	0

Table 2
Order-4 Exp-Golomb codes.

M	Codeword	Quantization index range
0	$1x_3x_2x_1x_0$	0–15
1	$01x_4x_3x_2x_1x_0$	16–47
2	$001x_5x_4x_3x_2x_1x_0$	48–111
...	...	...

Table 3
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
Frame rate	60

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).

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

As h_I is the upsampled version of h_{APIF} with zero-insertion, $h_I(k,l)$ in (7) and (10) is a non-zero coefficient only if the indices k and l are multiples of 4.

The proposed h_{APIF} has 7 × 7 taps, because it is the best tradeoff between the overhead size and the R-D performance. In Table 1, each cell represents a coefficient of h_{APIF}, labeled with the coding order. There are 25 coefficients to be coded, as the point symmetry is assumed, which means the coefficients have even symmetry with respect to the center point after raster scanning [10]. APIF coefficients, highly correlated in successive frames, are first temporally predicted. Then, the prediction errors, i.e., the differences between the APIF coefficients in the current and previous frames, are uniformly quantized to 2^{12} steps, which is considered precise enough, because any higher precision will not further improve the performance according to our tests. The quantization index of each APIF coefficients is coded using order-4 Exp-Golomb codes. Order-k Exp-Golomb codes have a generic form of [M zeros][1][INFO], where INFO is an $(M + k)$-bit field carrying information. As shown in Table 2 [14], a quantization index is first coded using $(M + k)$-bit fixed-length coding, where M is determined by the quantization index's range, and then the leading [M zeros][1] is added as the prefix to form the whole codeword. Exp-Golomb codes with larger orders favor flatter-shaped probability distribution function (pdf), and based on our study, order-4 Exp-Golomb codes well fit the pdf of the temporal prediction error of the APIF coefficients.

3. Experimental results

The proposed APIF is integrated into the VCEG's reference software KTA2.6 and is compared to 2-D non-separable AIF, frame-based 7 × 7-tap ALF, and the joint use of them, which means AIF and ALF are both enabled in KTA2.6. These three benchmarks are subsequently referred to as AIF, ALF, and AIF + ALF, respectively.

Table 4
Performance improvements in IPPP-coded sequences.

HD Sequences	AIF	ALF	BETTER(AIF, ALF)	AIF + ALF	APIF	E-AIF
Bigships	−10.31	−6.62	−10.31	−9.85	−9.89	−9.91
City	−14.67	−17.16	−17.16	−18.66	−20.84	−16.23
Crew	−22.84	−14.54	−22.84	−23.63	−22.07	−22.97
Harbour	−11.36	−12.44	−12.44	−13.79	−12.34	−10.57
Jets	−12.73	−7.96	−12.73	−12.71	−13.77	−11.70
Optis	−6.47	−5.00	−6.47	−7.05	−7.24	−6.61
Raven	−19.04	−18.41	−19.04	−20.24	−23.67	−20.17
Sailormen	−10.09	−10.48	−10.48	−12.68	−11.53	−10.69
Sheriff	−8.97	−8.47	−8.97	−9.72	−10.68	−9.86
ShuttleStart	−14.71	−10.65	−14.71	−14.41	−17.95	−17.03
Average	−13.12	−11.17	−13.52	−14.27	−15.00	−13.57

3.1. Rate-distortion performance

Table 3 gives the test conditions; Tables 4 and 5 show the coding gain compared with H.264/AVC High Profile, measured by the bit-rate reduction at the same PSNR or by the PSNR gain at the same bit-rate [15]. The averages over all the test sequences are shown in the bottom row. The analysis below is based on the IBBP sequence structure. Similar observations can be obtained based on the IPPP sequence structure.

APIF provides up to 8% more bit-rate reduction compared with AIF (see *City*) and up to 10% more bit-rate reduction compared with ALF (see *Crew*). On average, APIF outperforms either AIF or ALF by

Table 5
Performance improvements in IBBP-coded sequences.

HD Sequences	AIF	ALF	BETTER(AIF, ALF)	AIF + ALF	APIF	E-AIF
Bigships	−5.31	−5.98	−5.98	−7.29	−7.85	−5.94
City	−9.96	−14.46	−14.46	−15.51	−17.63	−12.14
Crew	−21.36	−9.75	−21.36	−22.88	−19.86	−21.83
Harbour	−10.16	−13.54	−13.54	−15.10	−13.19	−8.89
Jets	−6.34	−4.79	−6.34	−6.86	−9.15	−7.97
Optis	−5.25	−5.65	−5.65	−6.84	−6.49	−5.31
Raven	−13.26	−13.20	−13.26	−15.78	−17.84	−13.76
Sailormen	−5.30	−7.90	−7.90	−9.26	−8.05	−5.42
Sheriff	−5.39	−7.12	−7.12	−7.79	−8.00	−5.79
ShuttleStart	−7.57	−8.40	−8.40	−8.89	−10.65	−10.62
Average	−8.99	−9.08	−10.40	−11.62	−11.87	−9.77

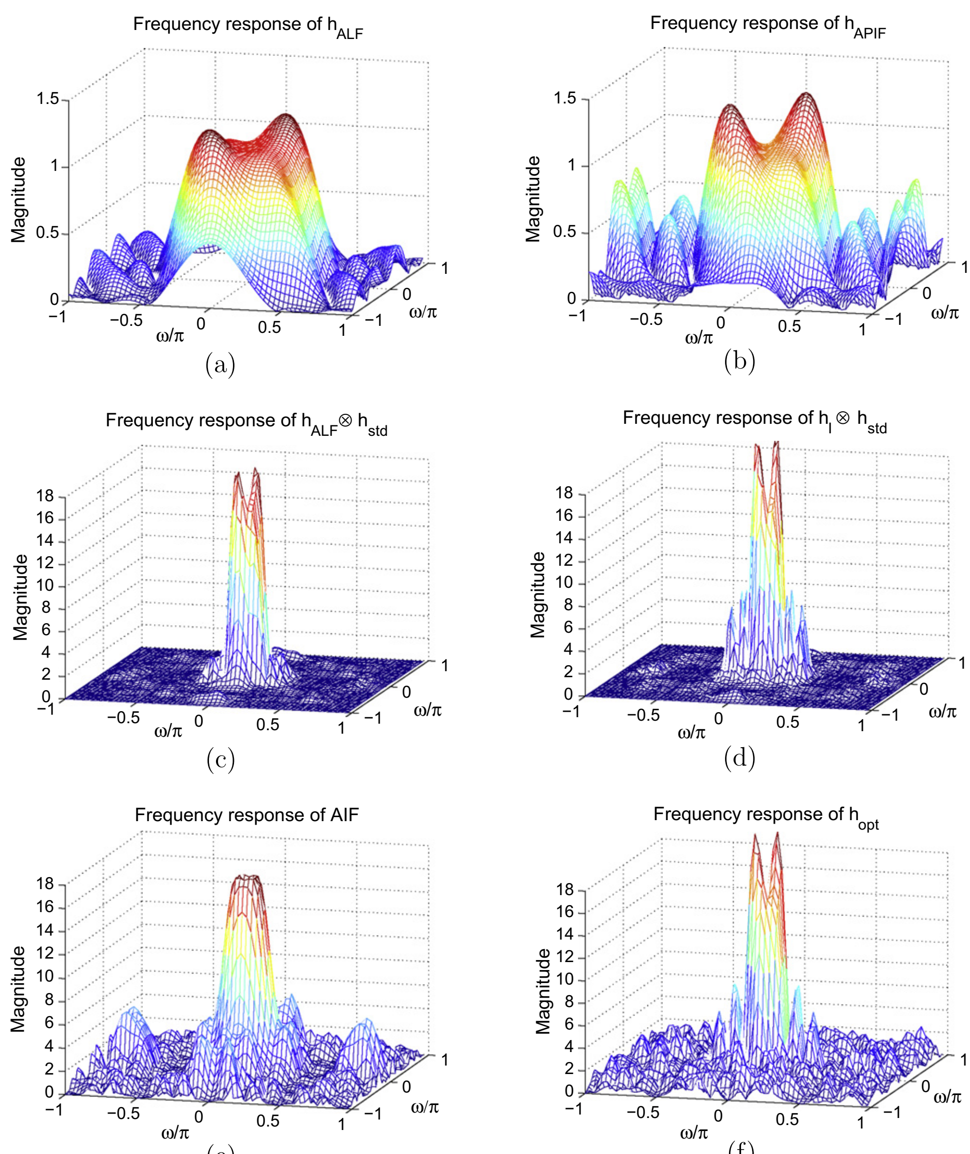


Fig. 4. In coding the 2nd *P*-frame of *Raven* (*QP* = 28), the frequency responses of (a) ALF, (b) APIF, (c) the convolution of ALF and h_{std}, (d) the convolution of APIF and h_{std}, (e) AIF, and (f) h_{opt}.

2.8% more bit-rate reduction. We also study the worst case that APIF is 0.04 dB less efficient than AIF in coding *Crew*. At high bit-rates (PSNR larger than 39 dB), APIF is slightly better than AIF. At low bit-rates, the optimal filter h_{opt} for each frame is not a normal low-pass filter; instead, it contains random high frequency pass bands. AIF represents such random pass bands better than APIF, because its filter coefficients are designed individually. As for APIF, the low-pass filter h_{std} (see Fig. 2(a)) filters out all the high frequency components.

It is observed that for some sequences AIF does not perform so well as ALF, and vice versa. For example, AIF outperforms ALF in coding *Crew* and *Jets*, whereas ALF outperforms AIF in coding *City*, and *Harbour*. The problem is that replacing one of AIF and ALF with the other will cause potential loss. APIF does not have the problem. For example, when coding *Crew*, where AIF outperforms ALF, the performance of APIF is comparable to that of AIF (we consider the performance gap less than 0.05 dB comparable); when coding *City*, where ALF outperforms AIF, APIF is even better than ALF. For each sequence, the better of the performances achieved by AIF and ALF is shown in the 6th and 7th columns, referred to as BETTER(AIF, ALF). For all the sequences, APIF's performances are always comparable to or better than BETTER(AIF, ALF). On average, 1.5% more bit-rate reduction is observed.

Furthermore, we compare APIF with the joint use of AIF and ALF. AIF + ALF provides 1.5% further bit-rate reduction on average compared to either AIF or ALF, whereas the complexities of AIF and ALF are additive. APIF, as a single coding tool, outperforms AIF + ALF in coding *City*, *Jets*, *Raven*, and *ShuttleStart*, where the best case is 0.1 dB PSNR improvement (see *Raven*). In coding *Bigships*, *Optis*, *Sailormen*, and *Sheriff*, the performances of APIF and AIF + ALF are comparable. It is noticed that APIF is worse than AIF + ALF in coding *Crew* and *Harbour*. For *Crew*, AIF has already been slightly better than APIF (The reason has been explained above); for *Harbour*, the performances of ALF and APIF are almost the same. Therefore, an additional AIF/ALF will make AIF + ALF outperform APIF. On average, APIF and AIF + ALF can be considered to have the same coding efficiency.

We also provide the performance of E-AIF [7], which reduces the support region of AIF, but loosens the symmetry assumption from isotropic to axial symmetry. E-AIF adds a 5×5 filter for integer pixels and a filter offset to each integer and sub-position pixel. E-AIF performs slightly better than AIF, but still cannot replace ALF without any loss. On average, APIF outperforms E-AIF with 2% and 1.5% more bit-rate reductions in IBBP- and IPPP-coded sequences, respectively.

Fig. 4 gives an example to demonstrate that APIF better approximates h_{opt} than other adaptive filters. The six filters shown in Fig. 4 are used for coding the second *P*-frame of *Raven* ($QP = 28$). As the optimal AIF h_{opt} (see its frequency response in Fig. 4(f)) can reduce the MCP error most, other filters with frequency responses resembling Fig. 4(f) more are considered more efficient. The frequency response of AIF (see Fig. 4(e)) is quite different from Fig. 4(f), in either the passband or the stopband. The frequency responses of ALF and APIF are shown in Fig. 4(a) and (b), respectively. Since these two filters are applied to integer pixels only, their capabilities of reducing MCP cannot be shown unless concatenated with the interpolation filter in H.264/AVC. Fig. 4(c) and (d) show the frequency responses of $h_{ALF} \otimes h_{std}$ and $h_I \otimes h_{std}$, respectively. Obviously, Fig. 4(d) resembles Fig. 4(f) more than Fig. 4(c), which means APIF is more efficient in reducing MCP error than ALF.

3.2. Complexity analysis

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 and E-AIF. Other factors, e.g., the number of equations to solve for LMMSE estimator, also influence the encoder complexity, but are relatively trivial. In short, the complexity of APIF is similar to AIF techniques, but is higher than frame-based ALF techniques, which employ the one-pass encoding.

For a decoder, no derivation process for filter coefficients is needed, since the coefficients, received from the bitstream, are used for filtering directly. Therefore, only the operations used directly 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. 5, according to the order the pixels are generated. The full-pixels are first filtered, of which the outputs are used to support interpolating half-pixel I. Then,

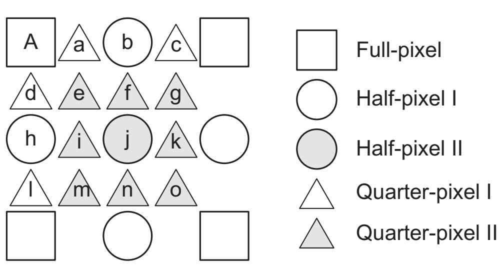


Fig. 5. Five categories of pixels to be filtered or interpolated.

Table 6
Arithmetic operations for interpolating one frame.

Filter	Filter in H.264		ALF		AIF	
Operation	Multiply	Add	Multiply	Add	Multiply	Add
Full-pixel	0	0	$49X/K$	$48X/K$	0	0
Half-pixel I	$6X$	$5X$	$6X$	$5X$	$6X$	$5X$
Half-pixel II	$6X$	$5X$	$6X$	$5X$	$36X$	$35X$
Quarter-pixel I	0	X	0	X	$6X$	$5X$
Quarter-pixel II	0	X	0	X	$36X$	$35X$
Total	$18X$	$27X$	$(49/K + 18)X$	$(48/K + 27)X$	$360X$	$345X$

Filter	E-AIF		AIF + ALF		APIF	
Operation	Multiply	Add	Multiply	Add	Multiply	Add
Full-pixel	$25X$	$25X$	$49X/K$	$48X/K$	$49X$	$48X$
Half-pixel I	$6X$	$6X$	$6X$	$5X$	$6X$	$5X$
Half-pixel II	$12X$	$12X$	$36X$	$35X$	$6X$	$5X$
Quarter-pixel I	$6X$	$6X$	$6X$	$5X$	0	X
Quarter-pixel II	$12X$	$12X$	$36X$	$35X$	0	X
Total	$169X$	$169X$	$(49/K + 360)X$	$(48/K + 345)X$	$67X$	$75X$

half-pixel II are interpolated, supported by the full-pixels and half-pixel I, and so on. Second, we calculate the required numbers of multiplications and additions for interpolating each category of pixels in an entire frame, as shown in Table 6, 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 6) can be calculated by (11),

$$N_{Total} = N_{Int} + 2N_{HalfI} + N_{HalfII} + 4N_{QuarI} + 8N_{QuarII} \tag{11}$$

where N_{Int}, N_{HalfI}, N_{HalfII}, N_{QuarI}, and N_{QuarII} are the operation numbers (multiplication or addition) for the five categories of pixels, respectively. Clearly, APIF has much lower complexity than AIF, AIF + ALF, and E-AIF, but is of more complex than ALF. The extent to which APIF is more complex than ALF is influenced by the number of reference frames K. Assuming a high complexity encoding configuration, where K is four, APIF doubles the complexity of ALF. When K is two for a moderate configuration, APIF has 1.5 times the complexity of ALF. However, when K reduces to one, the complexities of APIF and ALF become the same.

4. Conclusion

The paper proposes an interpolation filter comprising two concatenating filters: APIF and the interpolation filter in H.264/AVC. APIF, applied only to integer pixels in the reference frames, is designed such that the convolution of APIF and the standard filter minimizes the MCP error on a frame basis. APIF preserves the merits of AIF and ALF and at the same time overcomes their drawbacks. The experimental results show that APIF outperforms either AIF or ALF. Compared with the joint use of AIF and ALF, APIF provides comparable performance, but has much lower complexity.

References

[1] Vision and requirements for high-performance video coding (HVC), ISO/IEC JTC1/SC29/WG11 document N10361, 2009. [Online]. Available from: <http://wg11.sc29.org/meetings/87_Lausanne/w10361/w10361.zip>.

[2] Joint call for proposals on video compression technology, ITU-T Q.6/SG16 (VCEG) document VCEG-AM91, January 2010. [Online]. Available from: <http://wftp3.itu.int/av-arch/video-site/1001_Kyo/VCEG-AM91.zip>.

[3] Key Technical Area (KTA). [Online]. Available from: <http://iphome.hhi.de/suehring/tml/download/KTA>.

[4] H.264/AVC Test Model JM11.0. [Online]. Available from: <http://iphome.hhi.de/suehring/tml/download/old_jm/jm11.0.zip>.

[5] Y. Vatis, J. Ostermann, Adaptive interpolation filter for H.264/AVC, IEEE Trans. Circuits Syst. Video Technol. 19 (February) (2009) 179–192.

[6] D. Rusanovskyy, K. Ugur, A. Hallapuro, J. Lainema, M. Gabbouj, Video coding with low-complexity directional adaptive interpolation filters, IEEE Trans. Circuits Syst. Video Technol. 19 (August) (2009) 1239–1243.

[7] Improved filter selection for B-slices in E-AIF, ITU-T Q.6/SG16 (VCEG) document VCEG-AI38, July 2008. [Online]. Available from: <http://wftp3.itu.int/av-arch/video-site/0807_Ber/VCEG-AI38.zip>.

[8] Adaptive (wiener) filter for video compression, ITU-T Q.6/SG16 (VCEG) document COM16-C437, April 2008. [Online]. Available from: <http://www.itu.int/md/T05-SG16-C-0437/en>.

[9] Adaptive loop filter for improving coding efficiency, ITU-T Q.6/SG16 (VCEG) document COM16-C402, April 2008. [Online]. Available from: <http://www.itu.int/md/T05-SG16-C-0402/en>.

[10] Block-based adaptive loop filter, ITU-T Q.6/SG16 (VCEG) document VCEG-AI18, July 2008. [Online]. Available from: <http://wftp3.itu.int/av-arch/video-site/0807_Ber/VCEG-AI18.zip>.

[11] Quadtree-based adaptive loop filter, ITU-T Q.6/SG16 (VCEG) document COM16-C181, January 2009. [Online]. Available from: <http://www.itu.int/md/T09-SG16-C-0181/en>.

[12] J. Dong, K.N. Ngan, Parametric interpolation filter for motion compensated prediction, in: IEEE International Conference on Image Processing '09 (ICIP09), Cairo, Egypt, November 2009.

[13] J. Dong, K.N. Ngan, Parametric interpolation filter for high-definition video coding, in: IEEE Transactions on Circuits and Systems for Video Technology, in press.

[14] L. Zhang, Q. Wang, N. Zhang, D. Zhao, X. Wu, W. Gao, Context-based entropy coding in AVS video coding standard, Signal Process.: Image Commun. 24 (April) (2009) 263–276.

[15] G. Bjontegaard, Calculation of average PSNR differences between RD-curves, in ITU-T SG16/Q6, VCEG-M33, April 2001. [Online]. Available from: <http://wftp3.itu.int/av-arch/video-site/0104_Aus/VCEG-M33.doc>.