

# Efficient Implementation of Adaptive Interpolation Filters for Low Complexity Video Coding

K. Ugur, *Member, IEEE*, D. Rusanovskyy, *Member, IEEE*, A. Hallapuro, J. Lainema, M. Gabbouj, *Senior Member, IEEE*

**Abstract** — *This paper presents novel methods for implementing adaptive interpolation filtering techniques for video coding on 16-bit integer architectures. The proposed methods offer significant complexity reduction over the state-of-the-art, making them especially valuable for resource constrained mobile multimedia devices, with high coding efficiency<sup>1</sup>.*

**Index Terms** — Adaptive interpolation, video coding, interpolation filter.

## I. INTRODUCTION

Motion Compensated Prediction (MCP) is a technique used by all popular video coding standards to exploit the temporal redundancy present in a video sequence. In MCP, each frame of a video sequence is divided into blocks and a reference frame is searched for each block to find its best match. The relative position of the best matching block with respect to the original one is called the block's motion vector. Motion Vectors may have either integer or fractional pixel accuracies. If the motion vector has fractional pixel accuracy, the decoder needs to perform interpolation to obtain the samples at sub-pixel positions. The state-of-the art video coding standard, H.264/AVC, supports motion vectors with half and quarter pixel accuracies. The half-pixel samples are obtained by using a 6-tap filter and quarter-pixel samples are obtained by bi-linear filtering using the two nearest samples at half or integer pixel positions [1].

The interpolation filter defined in H.264/AVC was designed to minimize the adverse effects of aliasing present in the input image sequence [2]. However, aliasing in a video sequence is not a stationary process, but has a varying characteristic. Adaptive interpolation filters that change the filter coefficients at each frame have been proposed in the literature to combat this non-stationary effect of aliasing [3-5]. Because the coefficients of the adaptive filter can take any value and thus have high dynamic range, the filtering process requires 32-bit floating point registers. This result in a serious

complexity increase for adaptive interpolation schemes compared to 16-bit integer arithmetic implementation of the H.264/AVC interpolation filter. In our previous work [5], we presented adaptive interpolation scheme utilizing directional filters and its 16-bit integer implementation. These schemes were adopted to the reference software maintained by the ITU-T VCEG standardization group.

In this paper, we present a more detailed description of the 16-bit implementation of adaptive interpolation filters and extend it by including novel tools. The proposed algorithms achieve 16-bit integer implementation by restricting the dynamic range of the filter coefficients. The filter coefficients for different sub-pixels are defined with a variable accuracy by exploiting the different statistical distributions of the filter coefficients. In addition, a novel structure is proposed that utilizes partial convolution sums to gain additional accuracy and improve the coding efficiency further. Finally, we propose an adaptive rounding scheme to decrease the rounding error of the filter coefficients. The proposed algorithms could work with any adaptive interpolation filtering structure, but as an example we implemented them on our previously proposed Directional Adaptive Interpolation Filtering (DAIF) structure. Experimental results show that the proposed method achieves 16-bit operation with no penalty on coding efficiency compared to the 32-bit floating point implementation. When compared to H.264/AVC, the proposed methods achieve up to 19% bitrate reduction, with comparable implementation complexity.

This paper is organized as follows; Section 2 provides a brief background of adaptive interpolation filters and the DAIF structure. Section 3 presents the details of the proposed tools to achieve low complexity 16-bit implementation. Section 4 presents detailed theoretical analysis on the effect of the proposed tools on interpolation quality. Section 5 presents experimental results and detailed complexity analysis. Section 6 concludes the paper.

## II. ADAPTIVE INTERPOLATION FILTERS FOR VIDEO CODING

Consider Fig. 1.a., where the pixels at integer pixels are labeled by upper-case letters {A1,...,F6} within shaded boxes, and lower-case symbols {a,b,...,o} represent sub-pixels locations that are to be interpolated. In H.264/AVC, the sub-pixels at half-pixel positions (denoted with  $b$ ,  $h$ ,  $j$ ) are obtained using a 6-tap filter. The quarter-pixels are obtained by averaging the nearest half or full-pixel samples [1].

<sup>1</sup> This work was supported by Nokia Research Center and the Academy of Finland, (application number 213462, Finnish Programme for Centres of Excellence in Research 2006-2011)".

Kemal Ugur, Antti Hallapuro and Jani Lainema are with the Nokia Research Centre, Nokia Corp., Tampere, Finland. (e-mail: firstname.lastname@nokia.com)

Dmytro Rusanovskyy and Moncef Gabbouj are with the Department of Signal Processing at Tampere University of Technology, Tampere, Finland. (e-mail: firstname.lastname@tut.fi).

In the adaptive interpolation schemes, the encoder calculates the optimal filter coefficients that minimize the prediction error energy for each frame and transmits them to the decoder with the rest of the frame data. The calculations of the filter coefficients are performed as follows. Firstly, the frame is encoded with a fixed filter, such as the one defined in H.264/AVC standard and the initial motion vectors are found. Using these initial motion vectors and the reference frames, the interpolation filter coefficients for each sub-pixel are calculated analytically by minimizing the prediction error energy. Finally, the reference frame is interpolated using the optimal filter and the current frame is coded. In order to reduce the number of bits used to code the filter coefficients, it is assumed that statistical properties of the image signal are symmetrical. With this assumption, the filter coefficients are assumed to be the same, in case the distance of the corresponding full-pixel to the current sub-pixel is equal. For more details on calculating the filter coefficients and filter symmetry, the reader is referred to [3]

*A. Directional Adaptive Interpolation*

In [3], Vatis *et al.* proposed the use of a 2D non-separable adaptive interpolation filter to reduce the prediction error energy and improve the coding efficiency of video coders. For each sub-pixel position that is not aligned with integer pixels (sub-pixel positions  $e, f, g, i, j, k, m, n, o$ ), this scheme utilizes an 2D non-separable filter having a size of 6x6 taps. However, the improvement in the coding efficiency comes with the expense of having approximately three times higher interpolation complexity compared to the standard H.264/AVC [6]. The reasons for such complexity increase are the non-separable nature of 36-tap filters and the need to perform operations in 32-bits arithmetic. Since interpolation is already one of the most complex parts of H.264/AVC decoders, a further increase in the complexity is very problematic, especially for resource constrained devices, such as portable video players or recorders.

In [5], we proposed the Directional Adaptive Interpolation Filtering (DAIF) scheme to reduce the complexity of adaptive interpolation filtering. Instead of using 2D non-separable filters, DAIF structure uses directional filters, where the direction of the interpolation filter for different sub-pixel locations is determined based on the alignment of the corresponding sub-pixel with integer pixel samples. For example, consider Figure 1.a. and Figure 1.b. where the locations of the integer samples that are used in the filtering process are denoted by  $e, o$  and  $g, m$ ; respectively. The sub-pixel locations  $e, o$  are diagonally aligned with integer-pixels in the North-West—South-East direction. The integer-pixel positions that are aligned with these sub-pixels are  $\{A1, B2, C3, D4, E5, F6\}$ , thus only these integer-pixels are utilized to estimate  $h(e)$  and  $h(o)$  and to interpolate sub-pixels at locations  $e$  and  $o$ . Similarly, sub-pixel locations  $g, m$  are diagonally aligned with integer-pixels in the North-East—South-West direction. Therefore, integer

samples  $\{A6, B5, C4, D3, E2, F1\}$  are used for interpolating at sub-pixel locations  $g$  and  $m$ . Sub-pixel locations  $\{a, b, c\}$  and  $\{d, h, l\}$  are horizontally and vertically aligned with integer pixels. In this case,  $\{C1, C2, \dots, C6\}$  is used for interpolating at  $\{a, b, c\}$  and  $\{A3, B3, \dots, F3\}$  are used for interpolating at  $\{d, h, l\}$ . Locations  $\{f, i, j, k, n\}$  are either aligned with integer-pixel samples in both diagonal directions (location  $j$ ), or have no integer-pixel samples to be aligned with (positions  $f, i, k, n$ ). For this reason,  $\{f, i, j, k, n\}$  are interpolated using the integer-pixels that lie on a diagonal cross structure.

Since most of the sub-pixels are obtained using a single 6-tap filtering and the rest are obtained using 12-taps the interpolation complexity of DAIF is significantly less than its counterparts.

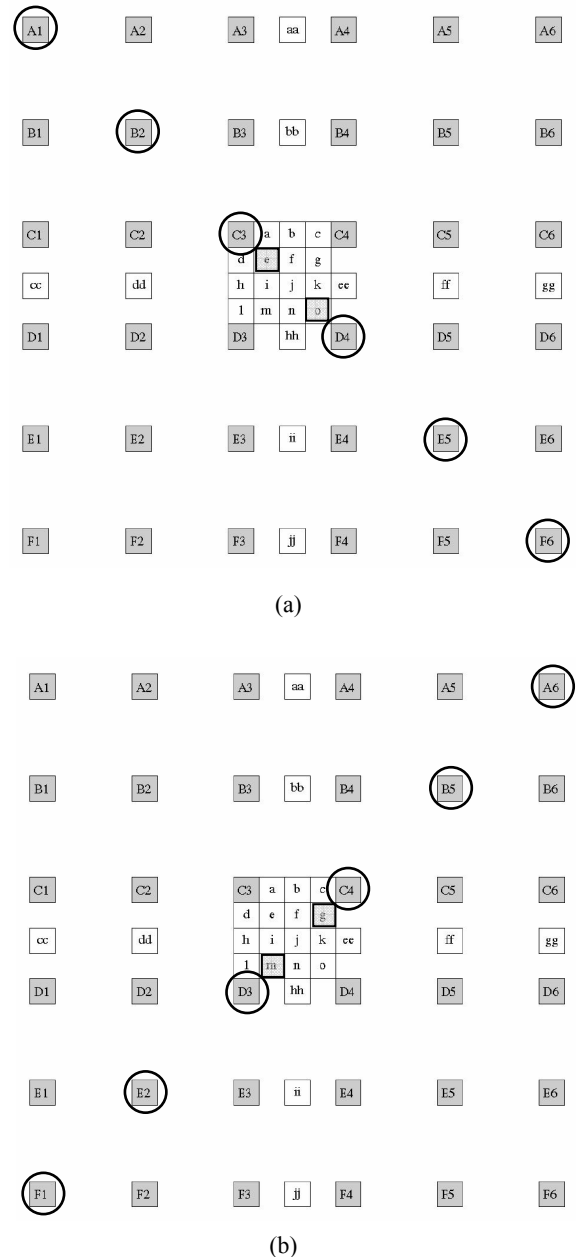


Fig. 1. DAIF structure for a) sub-pixel locations  $e, o$  and (b)  $g, m$

### III. LOW COMPLEXITY ADAPTIVE INTERPOLATION FILTERING

Because the coefficients of the adaptive filter can take any value and thus have high dynamic range, the filtering process requires 32 bit arithmetic. This result in a serious complexity increase for adaptive interpolation schemes compared to 16-bit integer arithmetic implementation of the H.264/AVC interpolation filter. One straightforward way to achieve a 16-bit operation for adaptive interpolation is to define the filter coefficients using smaller number of bits and decrease their accuracy. However, this results in a significant coding efficiency penalty and diminishes all the benefits of adaptive interpolation.

To investigate this problem, let us assume a one-dimensional 6-tap filter with coefficients in floating point arithmetic given as  $h_i$ , where  $i$  is the coefficient index. The filter coefficients in integer form are written as  $H_i$ , where  $Q$  determines the accuracy for the coefficients:

$$H_i = \text{sign}(h_i) \cdot \lfloor |h_i| \cdot 2^Q + 0.5 \rfloor \quad (1)$$

,where  $\text{sign}()$  function returns -1 if the input is negative and +1 if the input is positive, and  $\lfloor \cdot \rfloor$  is the floor function returning the nearest integer smaller than the input value.

The filter output is computed as follows:

$$Sp = \left( \sum_{i=0}^5 Y_i \cdot H_i + 2^{Q-1} \right) \gg Q \quad (2)$$

Where  $Y_i$  are the integer samples inside the filtering mask and  $Sp$  is the sub-pixel sample to be interpolated. Larger  $Q$  improves the accuracy of filter coefficients representation but increases the dynamic range of the convolution sum in (2). In order to have the convolution sum in (2) fit in a 16-bit register, the filter coefficients should be represented with maximum of 4 bits accuracy ( $Q=4$ ). It was found empirically, that having  $Q$  less than 8 significantly reduces the coding efficiency of adaptive interpolation schemes. The following subsections describe the proposed methods in order to increase the accuracy of the filter coefficients representation and thus preserve the coding efficiency of the adaptive interpolation scheme.

#### A. Partial Convolution Sums

The proposed structure performs interpolation not using equation in (2) above, but partial sums. Let us rewrite Equation (2) as:

$$Sp = \left( \sum_{i=0}^2 Y_i \cdot H_i + \sum_{i=3}^5 Y_i \cdot H_i + 2^{Q-1} \right) \gg Q \quad (3)$$

Assume that the partial sums  $\sum_{i=0}^2 Y_i \cdot H_i$  and  $\sum_{i=3}^5 Y_i \cdot H_i$  are stored in 16-bit registers. The sign of the adaptive filter coefficients are unknown, thus the accumulation registers  $res1$  and  $res2$  should support signed number representation. However, for typical interpolation filter coefficients and data samples, these partial sums will always be positive. We can gain 1 bit accuracy for filter coefficients representation by clipping negative  $res1$  and  $res2$  (i.e. if they are negative, set

their values to zero) and performing the final addition using unsigned, instead of signed arithmetic. This process is illustrated in Figure 2.

In order to see how this process increases the accuracy of filter coefficients, assume that the filter coefficients are represented in integer form with 7 bits accuracy (i.e.  $Q = 7$ ). For this example, we consider the interpolation filter defined in the H.264/AVC, denoted with  $h_{AVC}$  and  $H_{AVC}$  for floating point and in integer form, respectively, and are given as:

$$h_{AVC} = [1, -5, 20, 20, -5, 1]/32$$

$$H_{AVC} = [4, -20, 80, 80, -20, 4]$$

In order to see if the 16-bit register used for the interpolation given in Eq. 2 overflows, we need to find out the maximal value the convolution sum could take before shifting and see if it fits in the 16-bit register. The maximal value happens when the image samples that are multiplied with positive filter coefficients are 255 (assuming an 8-bit image), and the image samples that are multiplied with negative filter coefficients are 0. In other words, the maximum value for the convolution sum happens for filter  $H_{AVC}$  when the image samples  $Y$  take the following values:

$$Y = [255, 0, 255, 255, 0, 255]$$

In this case, the convolution sum  $\sum_{i=0}^5 Y_i \cdot H_i + 2^{Q-1}$  takes the value 42904 and requires a 17-bit signed register (16 bits for magnitude and 1 bit for sign), and therefore does not fit into a 16-bit register.

For the same example, let us now use partial convolution sums and clipping the intermediate values as described above. Each partial convolution sum given in Eq. 3 would take the value 21420 and fits in a 16-bit signed register. As the final summation is done using unsigned addition, and the magnitude of 42904 could be defined in a 16-bit, the interpolation operation does not overflow the 16-bit registers.

It should be noted that clipping the partial sums below zero loses some information to gain additional bit accuracy for the coefficients. However, for typical interpolation filters and input samples, partial sums would very often be greater than zero and this clipping is exploiting this fact. More detailed analysis of the effect of this step on interpolation quality is given in Section 4.

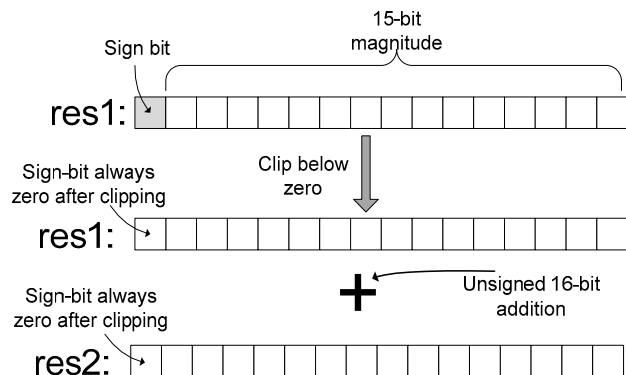


Fig. 2. Illustration of Intermediate Clipping

### B. Restriction on Filter Coefficients

In order for partial sums given in Eq. 3 to fit in 16-bit registers, we define two requirements on the filter coefficient values:

$$\left( \sum_{i=j}^{j+2} \max(0, H_i) < 128 \right) \text{AND} \left( \sum_{i=j}^{j+2} \min(0, H_i) > -128 \right) \quad (4)$$

where  $j$  is 0 and 3 for a 6-tap filter. With these restrictions in place, the accuracy of the filter coefficients is increased from 4 bits to 7 bits and still guaranteeing a 16-bit integer implementation. These requirements reduce the dynamic range of filter coefficients to ensure a 16-bit operation. Meanwhile, the dynamic range defined with the above restrictions cover typical interpolation filters used in video coding so that the coding efficiency is kept high. There are mainly two families of interpolation filters for which these requirements cannot be met. These are:

1. Interpolation filters with a gain larger than unity. Consider the case again of using the interpolation filter defined in H.264/AVC using 7 bits of accuracy. But instead of having unity gain, let's assume the gain is 2. The filter coefficients in integer form is given as:  
 $H_{AVC} = [8, -40, 160, 160, -40, 8]$   
 These filter coefficients do not meet the restrictions defined in Eq. 4. However, utilizing interpolation filters with a gain that is very different than unity in video coding happens in rare cases. And for those cases the video coder may choose to use different tools, such as Weighted Prediction [1].
2. Interpolation filters to obtain sub-pixels that are positioned very close to integer position. For these cases, the filter coefficients become highly asymmetrical and one of the filter coefficients would be significantly larger than the rest of the coefficients. This asymmetry in filter coefficients violates one of the restrictions given in Eq. 4. To illustrate this behavior, let's consider a widely used interpolation kernel that utilizes a 3-lobed Lanczos-windowed sinc function. This interpolation kernel is defined as:

$$\text{Lanczos3}(x) = \begin{cases} \frac{\sin(\pi x)}{\pi x} \cdot \frac{\sin(\pi \frac{x}{3})}{\pi \frac{x}{3}}, & |x| < 3 \\ 0, & |x| \geq 3 \end{cases} \quad (5)$$

where  $x$  is the distance between the sub-pixel and the integer samples. This kernel yields the following filter coefficients in a floating point representation to obtain a sub-pixel at position 92/100:

$$h = [0.0143, -0.0586, 0.9883, 0.0733, -0.0186, 0.0007]$$

$$H = [2, -8, 127, 9, -2, 0]$$

As seen above, the filter coefficients used to obtain a sub-pixel position very close to the integer sample does not fulfill the restriction given in Eq. 4.

For the rare cases, where these restrictions can not be met, the standard H.264/AVC interpolation filter coefficients are used for the adaptive filter. As it will be seen in Section 5 these restrictions do not deteriorate the performance of adaptive interpolation filters in video coding. The effect of reducing the dynamic range of the filter coefficients on interpolation quality is also analyzed further in Section 4.

### C. Filter Coefficients with Variable Accuracy

Additional bit accuracy for the filter coefficients is gained by observing the fact that the statistical distribution of the coefficients of different sub-pixels is different. If a sub-pixel is interpolated using a 6-tap filter, the statistical distribution of the coefficients would be different than that of the 12-tap interpolation filter. For example, the coefficients of a 12-tap filter most often fall in the range  $[-0.5, 0.5]$ , instead of  $[-1, 1]$ . We exploit this fact and restrict the filter coefficients of sub-pixel locations that use a higher number of taps. By using a restricted range, additional accuracy is gained for the respective sub-pixel locations.

### D. Adaptive Rounding of Filter Coefficients

The rounding operation given in Eq. 1 is not optimal in terms of minimizing the total rounding error of the interpolation filter. For some filters, the total rounding error accumulates to a large value, and hence penalizes the coding efficiency. To alleviate this effect, we add a correction factor to a number of filter coefficients after the rounding operation and reduce the total rounding error. This is done as follows. After the integer filter coefficients are computed using Equation 1, the total rounding error of the filter is found using:

$$\text{rnd\_error} = \sum_{i=0}^5 (h_i \cdot 2^i - H_i) \quad (6)$$

If the total rounding error is positive, this means that most of the coefficients were down-rounded. +1 is added to the coefficient that has the largest down-rounding error, then to the coefficient that has the second largest down-rounding error etc. This is repeated until the magnitude of the total rounding error falls below 0.5. Same process is done using the correction factor -1 for negative rounding errors. After this process, the modified coefficients have a higher rounding error, but the total rounding error of the filter is decreased.

In order to illustrate how adaptive rounding works, consider the following filter obtained using the Lanczos windowed sinc interpolation kernel, given in Eq. 5:

$$h = [0.0034, -0.0435, 0.1716, 0.9481, -0.1064, 0.0251]$$

For the case of regular rounding (rounding using Eq. 1), filter coefficients in integer form with a 7-bit accuracy are given as:

$$H = [0, -6, 22, 121, -14, 3]$$

Using Eq. 5, the total rounding error is 1.7897 and the rounding error for each coefficient is found as:

$$\text{rnd\_error}_H = [0.4379, 0.4356, -0.0413, 0.3579, 0.3872, 0.2125]$$

As seen above, the rounding error of all the coefficients is less than 0.5, but the total rounding error is 1.7897. Using the

above procedure, we first add +1 to the filter coefficient that has the largest down-rounding error, which is the first coefficient. Then the total rounding error decreases to 0.7897. We need to add +1 for one more filter coefficient that has the second largest down rounding error, and that is the second filter coefficient. This way the total rounding error becomes -0.2103. The modified filter coefficients and their respective rounding errors are then given as:

$$H' = [1, -5, 22, 121, -14, 3]$$

$$\text{rnd\_error}_{H'} = [-0.5621, -0.5644, -0.0413, 0.3579, 0.3872, 0.2125]$$

As seen clearly in the above example, the magnitude of the total rounding error of the filter is decreased from 1.7897 to 0.2103, even if some of the filter coefficients have a higher rounding error. The frequency response of the two rounding methods, rounding to nearest integer given in Eq. 1 and the proposed rounding as described above are presented in Fig. 3. As seen in Figure 3, rounding to the nearest integer value changes the frequency response of the filter significantly, but the frequency response comes closer to the original one when the proposed rounding is used. More important than the change in frequency response, adaptive rounding corrects the gain of the interpolation filter. In the above example, the original filter with floating point accuracy has unity gain; whereas filter  $h$  obtained using rounding to the nearest integer has a gain of 0.984375. In video coding, this change in gain means the encoder has to code more DC residual and degrade its coding efficiency. After adaptive rounding is applied, this gain is moved back to unity, which would help the video coder improve its coding efficiency.

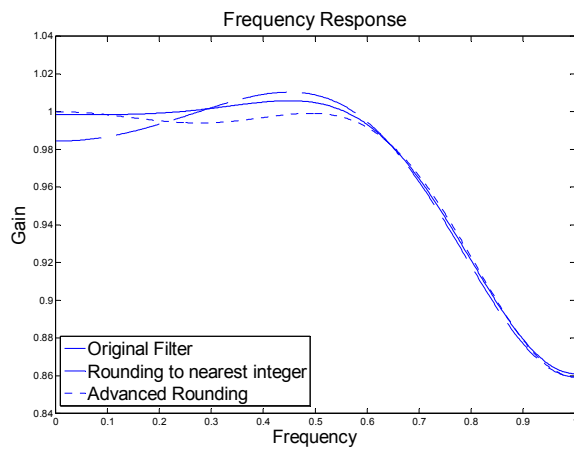


Fig. 3. Frequency response of filters with different rounding.

#### IV. EFFECT ON INTERPOLATION QUALITY

In this section, we analyze the effect of some of the proposed tools on the interpolation quality in more detail. The proposed tools are used in the context of video coding, where the filter coefficients are found at each frame by minimizing the prediction error energy. In order to simulate this process, we first define lanczos-windowed sinc function as the interpolation kernel. The coefficients of the adaptive interpolation filter are then given as a function of the sub-pixel

position using this interpolation kernel. As an example see Fig. 4, where the coefficients of the interpolation filter are given for the quarter-pixel position. We change the sub-pixel positions to vary the adaptive filter coefficients and analyze the effect of each proposed tool on this adaptive filter.

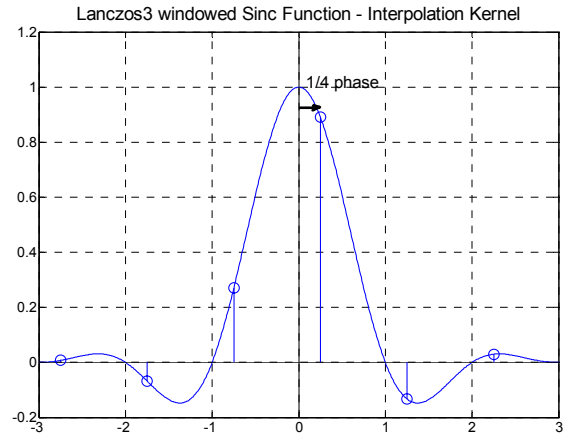


Fig. 4. Lanczos3 interpolation kernel and the respective filter coefficients for quarter-pixel position

##### A. Effect of restricting filter coefficients

We first analyze the effect of restricting the dynamic range of the filter coefficients using Eq. 4. This is done by utilizing different interpolation filters and checking if their respective coefficients satisfy the constraints given in Eq. 4. For this purpose, we use the lanczos-windowed sinc function given in Eq. 5, for sub-pixels whose position varies by increments of 1/100. In other words, we utilize 99 different interpolation filters corresponding to these sub-pixel positions located between two consecutive integer samples. If a high number of sub-pixel positions do not satisfy the constraints, it means that those sub-pixel positions can not be interpolated using the proposed tool. We also analyze how the gain of the interpolation filter changes the number of sub-pixel positions satisfying the constraints. The results of this experiment are given in Table 1.

Filter Gain	Number of filters not satisfying Equation 4
1	10: [2,3,4,7,8,92,93,96,97,98]/100
1.1	50
0.9	0

As seen in Table 1, for the unity gain, filters for the 10 sub-pixel positions out of 99 do not satisfy the constraints given in Eq. 4. As previously noted, those sub-pixel positions are the ones that are located very closely to an integer pixel. It should be noted that, in the context of video coding this much of fine-grain motion is not usually needed, and the fact that these sub-pixel positions do not satisfy the dynamic range requirements does not hinder the coding efficiency of a video coder. This is also shown in Section 5, where the proposed tools are integrated in a video coder.

### B. Effect of the proposed rounding

In order to analyze the effect of the proposed rounding scheme, we again use the same lanczos-windowed sinc function given in Eq. 5. Each sub-pixel position is interpolated first using floating point filter coefficients. Then the same sub-pixel position is interpolated using integer filter coefficients obtained with rounding to the nearest integer, as given in Eq. 1. Finally, the corresponding sub-pixel positions are interpolated using the filter coefficients obtained by the proposed rounding operation. The PSNR between the floating point interpolation and the two respective integer interpolations is plotted in Fig. 5.

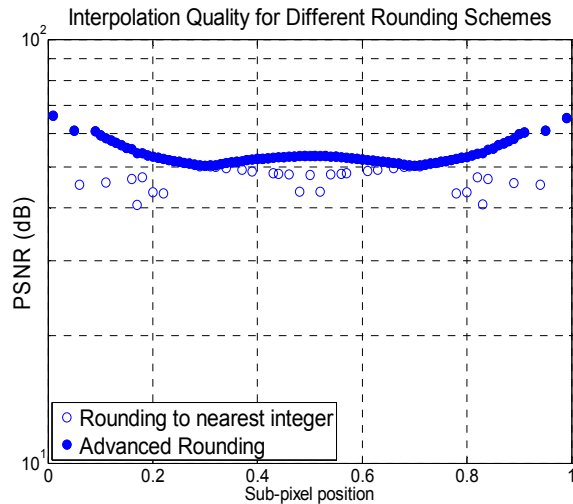


Fig. 5, Effect of rounding on interpolation quality

As seen in Fig. 5, the proposed rounding operation constantly improves the interpolation quality compared to rounding to the nearest integer scheme. For some sub-pixels the difference in PSNR is significantly high, and exceeds 1 dB.

## V. EXPERIMENTAL RESULTS

### A. Coding Efficiency Analysis

To evaluate the performance of the proposed algorithms in the context of video coding, we implemented them using the Directional Adaptive Interpolation Filtering structure [5] and performed extensive simulations. For the simulations, common conditions recommended by the ITU-T VCEG standardization group are used [7]. Table 2 summarizes the results for the Baseline profile settings, for 720p sequences. As seen in Table 2, the proposed methods achieve 16-bit operations with no penalty on coding efficiency, compared to 32-bit implementation. For some sequences, minor gains are also realized due to the adaptive rounding of the filter coefficients, and the lower dynamic range of filter coefficients that reduces the required number of bits to code them. When compared to H.264/AVC, the proposed methods achieve significant bitrate reduction (up to 19.39% gain).

TABLE 2  
CODING EFFICIENCY OF THE PROPOSED METHOD

Sequence	DAIF-16bits vs.	DAIF-16bit vs.
	H.264/AVC	DAIF-32 bit
	$\Delta$ Bitrate [%]	$\Delta$ Bitrate [%]
City	-16.51	-0.95
Bigships	-11.42	0.2
Crew	-19.39	-1.24
Shuttlestart	-19.31	-0.66
Night	-5.86	0.14

### B. Computational Complexity Analysis

In order to estimate the complexity of the proposed tools, we follow a similar methodology previously used to analyze the decoder complexity of H.264/AVC [8]. This methodology uses two basic steps to estimate the computational complexity of the interpolation filters. In step one, we count the number of arithmetic operations required to execute a filter for each sub-pixel. It is assumed that interpolation is done on a block-based and the number of arithmetic operations is counted for both 4x4 and 8x8 block sizes. In step two, we multiply the complexity numbers found in step one by the appropriate entry in a sub-function frequency table. The sub-function frequency table contains information regarding the use frequency of various sub-functions, the filters for each sub-sample location in this work, over the set of common condition sequences. The sub-function frequency table is generated by collecting the statistics for each sub-pixel location for both the H.264 and DAIF case using the Baseline profile's common conditions, as defined in [7]. To get more realistic numbers, it is assumed that motion blocks of size equal to or greater than 8x8 use the 8x8 interpolation whereas smaller block sizes use 4x4 interpolation.

Table 3 presents the number of arithmetic operations in 16-bits required to obtain each sub-pixel (i.e. the result of Step-1 as outline above). As seen in Table 3, the maximum increase in operations for DAIF-16-bit compared to H.264 interpolation occurs at locations b and h that are horizontally and vertically aligned half-pixel samples, respectively. On the other hand, diagonally aligned locations, e, g, m and o provide DAIF with a 43.5% computational advantage, as those locations are obtained using a single one-dimensional filter. On average, DAIF-16-bit has 21.7% less complexity than H.264 interpolation for the 4x4 case and 13.7% less complexity for the 8x8 case.

TABLE 3  
NUMBER OF 16-BIT ARITHMETIC OPERATIONS REQUIRED TO INTERPOLATE EACH SUB-PIXEL

Sub-pixel position	H.264/AVC		DAIF-16bit	
	4x4	8x8	4x4	8x8
b,h	10	10	17	10
a,c,d,l	13	13	17	13
e,g,m,o	23	23	17	17
j	32.5	26.25	35	35
f,i,k,n	35.5	29.25	35	35
Average for sub-pixels	22.56	20.48	23	23

When the complexity numbers of Table 3 are weighted with the appropriate entry in the sub-function frequency table, we arrive to a realistic estimate of complexity of the proposed tools, in which DAIF-16bits is 7% less complexity than H.264/AVC interpolation.

It should be noted that, the above analysis is based on counting the number of arithmetic operations and does not take into account several issues such as memory bandwidth, parallelism etc. A more detailed study analyzing the complexity of DAIF-16-bit, taking into account Single-Instruction-Multiple-Data (SIMD) parallelism, can be found in [9].

## VI. CONCLUSIONS

In this paper, novel algorithms are proposed that guarantee 16-bit integer implementation for adaptive interpolation filtering schemes with a high coding efficiency. When compared to H.264/AVC, the proposed methods achieve up to 19% bitrate reduction, with a lower computational complexity.

## REFERENCES

- [1] ITU-T and ISO/IEC JTC 1, "Advanced Video Coding for Generic Audiovisual Services," ITU-T Rec. H.264 & ISO/IEC 14496-10
- [2] T. Wedi and H. G. Musmann, "Motion- and Aliasing-Compensated Prediction for Hybrid Video Coding", IEEE Trans. on CSVT, vol. 13, No. 7, July 2003.
- [3] Y. Vatis, B. Edler, D. T. Nguyen, J. Ostermann, "Motion-and Aliasing-compensated Prediction using a Two-Dimensional Non-Separable Adaptive Wiener Interpolation Filter", in Proc. IEEE ICIP, Sept. 2005.
- [4] S. Wittman and T. Wedi, "Separable adaptive interpolation filter", ITU-T SG16/Q6, Doc. C-0219 July, 2007.
- [5] D. Rusanovskyy, K. Ugur, A. Hallapuro, J. Lainema, M. Gabbouj, "Video Coding with Low Complexity Directional Adaptive Interpolation Filters", *accepted*, IEEE Trans on CSVT.
- [6] Y. Vatis and J. Ostermann, "Comparison of complexity between twodimensional non-separable adaptive interpolation filter and standard wiener filter", ITU-T SGI 6/Q.6 Doc. VCEG-AA11, Nice, France, October 2005.
- [7] T.K Tan, G. Sullivan and T. Wedi, "Recommended Simulation Common Conditions for Coding Efficiency Experiments", ITU-T Q6/SG16, VCEG-AE010, Marrakech, Morocco, January 2007.
- [8] Complexity Analysis: H.264 vs. Computationally Efficient H.264, S. Perschau and M. Horowitz, ITU SG16 COM 16-D 140-E, presented in Geneva, CH, 26 July - 5 August 2005.
- [9] A. Hallapuro, K. Ugur, D. Rusanovskyy, J. Lainema, "Complexity Analysis: Directional Adaptive Interpolation Filters vs. H.264 Interpolation", ITU-T SGI 6/Q.6 Doc. C438, Geneva, Switzerland, April-May 2008.



**Kemal Ugur** received his M.Sc. degree in electrical and computer engineering from University of British Columbia, Canada in 2003 and joined Nokia Research Center in year 2004. Currently, he is working as a member of research staff and leading a project on next generation video coding technologies. Since he joined Nokia, he has been actively participating to several standardization forums, such as Joint Video Team (JVT) for the standardization of Multiview Video Coding extension of H.264/AVC, Video Coding Experts Group (VCEG) for exploration towards next generation video coding standard and 3GPP for mobile broadcast and multicast standard. He has more than 20 publications in academic conferences and journals and around 20 patent applications. In parallel, he is pursuing his PhD degree at Tampere University of Technology. He is a member of the research team that won the highly-prestigious Nokia Quality Award in the year 2006.



**Dmytro Rusanovskyy** received the Dipl.Eng. degree from National University of Radio Electronics, Kharkiv, Ukraine in 2000, and the M.Sc. degree in computer science from University of Jyväskylä, Finland in 2002. Currently, he is working as an external researcher in Nokia Research Center, and pursuing his PhD degree at Tampere University of Technology. His research interests include image, video processing and coding. He has been actively contributing to the work of ITU-T Video Coding Experts Group (VCEG) and he is an author/co-author around of 10 academic papers, several patent applications and adopted VCEG proposals.

**Antti Hallapuro** joined Nokia Research Center in 1998 where he has been working on video coding related topics. He has participated in H.264 video codec standardization and is author of several input documents. He has been involved in productization of H.264 codec in various forms. His interests are in practical video coding and processing algorithms and their high performance implementations.



**Jani Lainema** received the M.Sc. degree in computer science from the Tampere University of Technology, Finland in 1996. He joined the Visual Communications Laboratory of Nokia Research Center in 1996. Since then he has contributed to the designs of ITU-T's and MPEG's video coding standards as well as to the evolution of different multimedia service standards in 3GPP, DVB and DLNA. Recently he has been working as a Research Leader and Principal Member of Research Staff at Nokia Research Center Tampere. His research interests include video, image and graphics coding and communications, as well as practical applications of game theory.



**Moncef Gabbouj** (M'85-SM'95) received his BS degree in electrical engineering in 1985 from Oklahoma State University, Stillwater, and his MS and PhD degrees in electrical engineering from Purdue University, West Lafayette, Indiana, in 1986 and 1989, respectively. Dr. Gabbouj is a Professor at the Department of Signal Processing at Tampere University of Technology, Tampere, Finland. He was Head of the Department during 2002-2007. Dr. Gabbouj was a visiting professor at the American University of Sharjah, UAE, in 2007-2008 and Senior Research Fellow of the Academy of Finland in 1997-1998 and 2007-2008. His research interests include multimedia content-based analysis, indexing and retrieval; nonlinear signal and image processing and analysis; and video processing and coding.

Dr. Gabbouj served as Distinguished Lecturer for the IEEE Circuits and Systems Society in 2004-2005. He served as associate editor of the *IEEE Transactions on Image Processing*, and was guest editor of *Multimedia Tools and Applications*, the European journal *Applied Signal Processing*. He is the past chairman of the IEEE Finland Section, the IEEE CAS Society, Technical Committee on DSP, and the IEEE SP/CAS Finland Chapter.

Dr. Gabbouj was the recipient of the 2005 Nokia Foundation Recognition Award and co-recipient of the Myril B. Reed Best Paper Award from the 32nd Midwest Symposium on Circuits and Systems and co-recipient of the NORSIG 94 Best Paper Award from the 1994 Nordic Signal Processing Symposium. He is a member of IEEE SP and CAS societies.