Chapter 6

Low Complexity Interpolation Filters for Motion Estimation and Application to the H.264 Encoders

Georgios Georgis, George Lentaris and Dionysios Reisis

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/51703

1. Introduction

Techniques for image super-resolution play an important role in a plethora of applications, which include video compression and motion estimation. The detection of the fractional displacements among frames facilitates the removal of temporal redundancy and improves the video quality by 2-4 dB PSNR [12], [2]. However, the increased complexity of the Fractional Motion Estimation (FME) process adds a significant computational load to the encoder and sets constraints to real-time designs. Researchers have performed timing analysis for the motion estimation process and they reported that FME accounts for almost half of the entire motion estimation period, which in turn accounts for 60-90% of the total encoding time depending on the design configuration [12].

The FME bases on an interpolation procedure to increase the resolution of any frame region by generating sub-pixels between the original pixels. In mathematics, interpolation refers to the construction of an interpolant function whose plot covers (i.e. passes through) all required points. Known points of a sample area are referred to as having integer interval or displacement, depending on whether they are time or frequency-domain (TD or FD) samples respectively. Similarly, unknown samples which have to be approximated through an interpolant function, are said to have fractional interval or displacement respectively. In images, the interpolation takes place in a two-dimensional frequency-domain grid, where the problem of calculating fractional displacements can be facilitated by focusing on an area of four initially known pixels which reside on the corners of a unit square (Fig. 1). Hence, regardless of the interpolation factor, it is adequate to calculate pixels with arbitrary displacements in the unit square and extend the calculation for every unit square, which belongs to the frame.

Most of the non-adaptive techniques presented in the bibliography, base on solving piecewise polynomial functions of varying degrees in order to calculate the interpolated signal. The resulting polynomial solution leads to sets of coefficients to be applied on consecutive sample points in the grid, which most often extend beyond the unit square. Examples of the above approach are first, the Bilinear interpolation [8] with first order polynomials and using two pixels in each dimension and second, the Bicubic interpolation [9] which is derived from third order polynomials and uses four pixels in each dimension. On the other hand, Lanczos interpolation coefficients [10] stem from windowing a *sinc* function. Therefore, the number of pixels required by the Lanczos approach depends on the choice of the order of the interpolation function. More complex techniques applied to video encoding, employ edge-detection, error function minimization, or super-resolution (SR) procedures originating from theoretical signal processing methods. Among these techniques, the most commonly used is the edge-detection, which characterizes pixels or areas in an image belonging to an edge (luminance inconsistency). Edge-detection is also utilized for preventing aliasing frequency components to be encoded and transmitted.

Modern compression standards specify the exact filter to use in the Motion Compensation module, a fact allowing the encoder and the decoder to create and use identical reference frames. In particular, H.264/AVC specifies a 6-tap filter for generating sub-pixels between the pixels of the original image, which are called half-pixels with accuracy $\frac{1}{2}$ [3]. Also, it defines a low cost 2-tap interpolation filter for generating sub-pixels between half- and original pixels, which are defined as quarter-pixels with accuracy $\frac{1}{4}$. Even though it is a common practice among the encoder designers to integrate the standard 6-tap filter also in the Estimation module (before Compensation), the fact is that the interpolation technique used for detecting the displacements (not computing their residual) is an open choice following certain performance trade-offs.

Aiming at speeding up the Estimation, a process of considerably higher computational demand than the Compensation, this chapter builds on the potential to implement a lower complexity interpolation technique instead of using the costly H.264 6-tap filter. For this purpose, we show the results of integrating in the Estimation module several distinct interpolation techniques not included in the H.264 standard. We keep the standard H.264/AVC Compensation and we measure the impact of the above techniques first on the time required to process the up-sampling and second on the video quality achieved by the prediction engine.

Related results in the bibliography include techniques, which lead to avoid or replace the standard computations [4] [5] [13], or minimize the search area [14]. Researchers in [4] calculate the number of operations required for each pixel in cases where 8-to-2-tap filters and the Sum of Absolute Differences (SAD) metric is utilized. Then, they perform statistical analysis in CIF sequences encoded using bitrates from 0.5 to 1Mbps, to determine the recurrence of a motion vector when the aforementioned filter lengths are applied. The authors of [5] and [13] initially focus on reducing the number of taps and the multiplication operations, by proposing a filter which requires only shifts and additions. Then they propose adaptive thresh-

http://dx.doi.org/10.5772/51703

olds to bypass the interpolation process based on the computed SAD value. Recent developments towards replacing the H.264 / AVC (High Efficiency Video Coding or H.265 or MPEG-H part 2) [16], combine Rate-Distortion minimization and adjustments to local image characteristics [15], [17], [18], [19]. Effectively, these techniques switch between standard and directionally adaptive interpolation kernels and they take this decision by examining each frame either on a pixel or macroblock basis.

Conventional Super-resolution (SR) techniques are generally considered to be prohibitively expensive when encoding video sequences. However, in many cases the learning-based super-resolution techniques are considered to be valid [20]. Consisting of a training phase, where low and high-resolution image patches are matched and a synthesis phase, where low resolution patches kept in the dictionary are used to oversample, learning-based SR provides increased PSNR whilst expanding storage and memory access requirements. Researchers and engineers have also focused on methodologies for designing the H.264 6-tap filter, which are able to efficiently support its increased memory requirements [2] [6] [7]. The H.264 filter needs quite a number of data to be stored for its operation because its specifications include a kernel with coefficients $\langle 1, -5, 20, 20, -5, 1 \rangle$, which are multiplied with six consecutive pixels of the frame either in column or row format. The resulting six products are accumulated and normalized for the generation of a single half-pixel, which is produced between the 3^{rd} and the 4^{th} tap. The operation described above must be repeated for producing each "horizontal" and "vertical" half-pixel by sliding the kernel on the frame, both in row and column order. Moreover, there exist as many "diagonal" half-pixels to be generated by applying the kernel on previously computed horizontal or vertical half-pixels. That is to say, depending on its position, we must process 6 or 36 frame pixels to compute a single half-pixel. To avoid the cost of implementing the H.264 filter in the Estimation module, the current chapter studies a set of interpolation techniques and compares their performance. The techniques presented here are similar to the standard filter but they use less than 6 taps [8] [9] [10]. Moreover, a subset of these techniques features the exploitation of gradients in the image [11].

The chapter is organized as follows: Section 2 shows three commonly used interpolation techniques, proposes three novel techniques and describes the differences among those commonly used and the proposed. Section III reports the performance results achieved by the interpolation techniques and by comparing these shows the gains of using the proposed. Finally, Section IV concludes the chapter.

2. Interpolation techniques

The current section presents six interpolation techniques. The first three (3) are known in the literature and are commonly used techniques. The other three (3) have been recently introduced [13] and their design targets the improvement of the interpolation process.

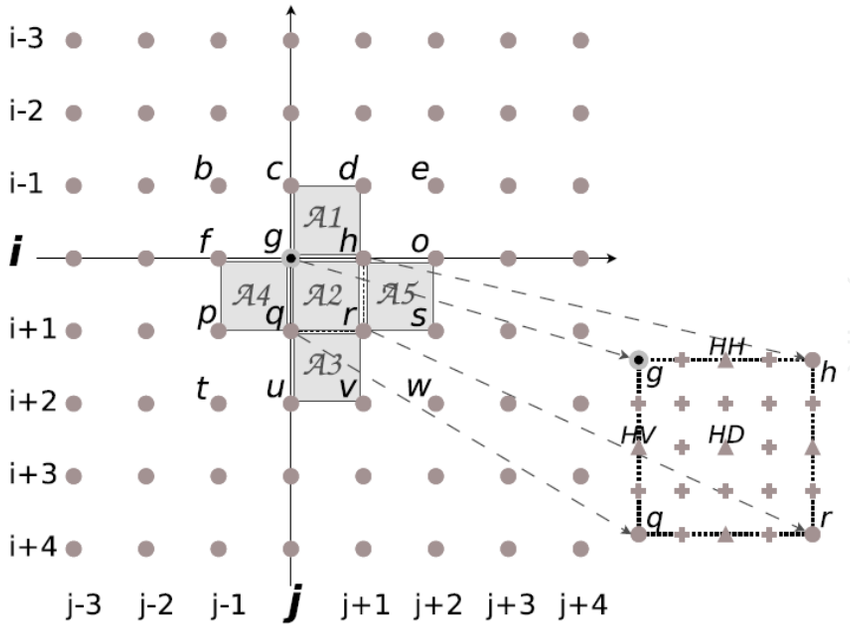

Figure 1. Pixels on the image grid and magnification of a 1×1 area showing sub-pixel positions (right). The symbols facilitate the description of filters.

Each video frame consists of pixels and we consider each pixel of the original image located at a distinct position (i, j) of a two dimensional (2D) grid with $i,\ j \in \mathrm{N}$ denoting the vertical and horizontal coordinates of the pixel, respectively. The sub-pixels can be generated next to any pixel (i, j) at the positions (i+k, j+l) with $k,l \in \left\{0,\ \frac{1}{4},\ \frac{1}{2},\ \frac{3}{4}\right\}$.

We distinguish between quarter-pixels and half-pixels, for which $k,l \notin \left\{\frac{1}{4},\ \frac{3}{4}\right\}$. The half-pixels are further categorized as half-horizontal, half-vertical, or half-diagonal (those located at the positions given by $\left(i+\frac{1}{2},\ j+\frac{1}{2}\right)$). Fig. 1 depicts part of the original image grid and magnifies a small area while the right-hand side magnifies an interior square region to show all sub-pixel positions (according to H.264/AVC). Moreover, Fig. 1 marks pixels and regions on the grid to be used as references with designated letters as a notation to be followed for the remaining of the paper.

A half-pixel is generated by an interpolation procedure operating on a set of neighboring, integer-position pixels located around the position of interest. We study the following interpolations:

2.1 Bilinear

This technique is actually the simplest of all the techniques presented in this chapter. In practice, this technique consists of a simple averaging of the two original pixels, which are adjacent to the half-horizontal or the half-vertical pixel to be generated (i.e., 2-tap FIR filter) [8]. For the half-diagonal (HD), the technique computes the average of the four (4) pixels {g, h, q, r} surrounding the half-diagonal position as shown in (Fig. 1).

http://dx.doi.org/10.5772/51703

2.2 Bicubic

The Bicubic technique uses as a base the solution of third order polynomials [9]. In this chapter we examine the parameterized form of the underlying equation using $a \in [-1, 0]$ to provide sharpness variance in the interpolated image. We focus on the following values: $a = -1$, $a = -0.75$, and $a = -0.5$. These values result in three distinct kernels, which are characterized by the convolution coefficients $\langle -1,5, 5, -1 \rangle$, $\langle -3,19,19, -3 \rangle$ and $\langle -1,9, 9, -1 \rangle$, respectively. Such a quadruplet is multiplied with four (4) consecutive image pixels to generate their intermediate half-pixel. To compute the half-diagonal pixel, the Bicubic technique requires the calculations of the corresponding four half-horizontal (a total of 16 multiplications) and then apply the coefficients on the resulting pixels to produce the target half-diagonal. Hence, overall it uses 16 image pixels with the requirement of 20 multiplications.

2.3 Lanczos

This technique is similar to the H.264/AVC interpolation and with a third order Lanczos equation, it uses a 6-tap FIR filter. Overall, the technique bases on the Sinc function [10]. In this chapter we examine the kernel with coefficients given by $\left\langle \frac{12}{50\pi^2}, -\frac{12}{9\pi^2}, \frac{6}{\pi^2}, \frac{6}{\pi^2}, -\frac{12}{9\pi^2}, \frac{12}{50\pi^2} \right\rangle$. Lanczos half-pixels are generated by a trivial convolution procedure, as in the case of the H.264/AVC filter (a single half-diagonal pixel depends on 36 integer pixels). Note here that, the H.264/AVC standard defines a 6-tap filter for use in motion compensation with coefficients $\langle 1, -5,20,20, -5,1 \rangle$.

2.4 Data-Dependent Triangulation

The first of the recently introduced techniques in [13] is actually a modification of the approach, which was presented in [11]. The authors in [11] use an edge-detection technique for determining the exact set of integer pixels, which will be given as input to the interpolation function. We study here a special case of Data-Dependent Triangulation (DDT), which examines only 4 pixels. To describe the technique, we consider the generation of the half-horizontal (HH) pixel Y_D^{HH} at $(i, j+\frac{1}{2})$ and the half-vertical (HV) pixel Y_D^{HV} at $(i+\frac{1}{2}, j)$ as shown in Fig. 1.

We examine the luma differences of pixels $\{g, h, q, r\}$ to determine whether an edge crosses their enclosed region: if it holds that $|Y_g - Y_r| > |Y_h - Y_q|$, then we will detect an edge at hq, else we will detect an edge at rg. In the first case, that is there is an edge at hq which is denoted as ${}_h^q E_D$, we assume that pixels $\{g, h, q\}$ form a homogeneous triangular and we compute:

$$\begin{aligned} Y_D^{HH} &= Clip_{div_D}^R\left(w_1 Y_g + w_1 Y_h + w_2 Y_q\right) \\ Y_D^{HV} &= Clip_{div_D}^R\left(w_1 Y_g + w_1 Y_q + w_2 Y_h\right) \end{aligned} \tag{1}$$

Where $Clip^{R}_{div_D}$ is a normalization function (divides by $div_D = 2w_1 + w_2$, clips value in [0, 255]). Factors $w_1 > w_2$ are used to increase the luma weights of the neighbors residing next to the generated sub-pixel. The examination of a large number of factors has resulted in highest PSNR for $w_1 = 7$ and $w_2 = 2$ (given that $div_D = 2^4$). The second case refers to the detection of an edge at *rg* (when there is the edge ${}^{g}_{r}E_D$). In this case, we use the same idea as above (orientation and weights) but we modify accordingly the luma inputs of (1). In the case of a homogeneous square *ghqr* the technique degenerates to a simple bilinear filter (i.e. $w_1 = 1$, $w_2 = 0$).

The technique generates the half-diagonal pixel by including a second gradient check, which follows the detection of the edge ${}^{q}_{h}E_D$, or the edge ${}^{g}_{r}E_D$. The idea is to identify the most homogeneous triangle in the enclosed area *A*2 shown in the Fig. 1. Thereby, in the case of ${}^{q}_{h}E_D$, we check $|Y_g - Y_q| + |Y_g - Y_h| < |Y_r - Y_q| + |Y_r - Y_h|$, otherwise we check $|Y_h - Y_g| + |Y_h - Y_r| < |Y_q - Y_g| + |Y_g - Y_r|$ to decide if the HD pixel resides *above* (<) or *below* (>) the edge. Extending our notation with *abv* and *blw* superscripts, we describe the modified DDT (mDDT) computation as:

$$Y_D^{HD} = \begin{cases} Clip^{R}_{dlv_D}\left(w_1 Y_h + w_1 Y_q + w_2 Y_g\right) \; if^{q}_{h} \; E_D^{abv} \\ Clip^{R}_{dlv_D}\left(w_1 Y_h + w_1 Y_q + w_2 Y_r\right) \; if^{q}_{h} \; E_D^{blw} \\ Clip^{R}_{dlv_D}\left(w_1 Y_h + w_1 Y_q + w_2 Y_h\right) \; if^{8}_{r} \; E_D^{abv} \\ Clip^{R}_{dlv_D}\left(w_1 Y_h + w_1 Y_q + w_2 Y_g\right) \; if^{q}_{h} \; E_D^{abv} \\ Clip^{R}_{4}\left(Y_q + Y_h + Y_r\right) \; if^{8}_{r} \; unif \end{cases} \quad (2)$$

Where the values of the w_1, w_2 and $Clip^{R}_{div_D}$ are as described in (1).

An alternative approach uses the equation 1 to develop a simpler HD generation technique, we call this technique *mDDT'*, which relies directly on the first DDT check and performs a bilinear operation on the two pixels of the detected edge, i.e., $Y^{HD}_{D'} = Clip^{R}_{2}(Y_h + Y_q)$ *if* ${}^{q}_{h}E_D$.

We further improve the *mDDT'* and produce the (*mDDT'*) technique by modifying the final operation to subtract the remaining two off-diagonal pixels (as a high-pass FIR), i.e., $Y^{HD}_{D''} = Clip^{R}_{D''}(w_1 Y_h + w_1 Y_q - w_2 Y_g - w_2 Y_r)$ *if* ${}^{q}_{h}E_D$. The latter operation although it increases the amount of calculations, it results in better PSNR compared to the *mDDT'*.

http://dx.doi.org/10.5772/51703

2.5 CrossHD

The second approach is called CrossHD [13] and bases on an edge-oriented technique. The advantages of CrossHD compared to the DDT mentioned above, is that it improves on the locality of the aforementioned DDT detections by comparing the luminance difference of areas –instead of single pixels. This technique computes the luma of a small square area by adding the pixels, which are located at its four corners. For instance, for the example given in Fig. 1, we get that: $Y_{A1}=Y_c+Y_D+Y_g+Y_h$. The technique examines the outcome of the $|Y_{A4}-Y_{A5}|>|Y_{A1}-Y_{A3}|$ operation to decide if there exists a vertical (>) or horizontal (<) edge crossing the area A_2 . In the case of a vertical edge crossing the area A_2 , we examine independently the areas A_1 , A_2 , and A_3 by using the simple DDT check to identify the directions of the edges crossing each of these three (3) areas. The majority of the edge directions found within A_1 , A_2 , and A_3 refines the assumed edge direction within A_2 , i.e., we conclude if ${}^{q}_{h}E_{\chi}$ or ${}^{g}_{r}E_{\chi}$. Note that, in the case of examining whether there exists a horizontal edge, the technique will examine the areas A_4 , A_2 , and A_5 . Finally, the HD pixel is generated by averaging the pixels, which reside on the detected edge: $Y_{\chi}^{HD}=Clip_2^R(Y_h+Y_q)\ if\ {}^{q}_{h}E_{\chi}$, or $Y_{\chi}^{HD}=Clip_2^R(Y_r+Y_g)\ if\ {}^{g}_{r}E_{\chi}$. If the technique does not detect any edge (i.e., in the homogeneous square A_2), it will average the pixels {*g*, *h*, *q*, *r*}.

2.6 CxScale

The third approach extends the aforementioned ideas to develop a technique called CxScale [13], which improves both the edge detection and the subsequent kernel selection. Here, the edge detection mechanism examines the luma gradients over an area of 8 neighboring integer pixels and the half-pixels are generated afterwards via a conditional use of bilinear and bicubic interpolators. The technique includes three steps:

1. The detection of a horizontal or vertical edge.
2. The possible refinement of its direction to an assumed diagonal.
3. The selection of inputs to a bicubic or a bilinear function.

The specifics of these steps depend on the position of the half-pixel to be generated. Beginning with the HH pixel, we examine $|(Y_f+Y_g)-(Y_h+Y_o)|<|(Y_c+Y_d)-(Y_q+Y_r)|$ to detect a horizontal edge, i.e. ${}^{h}_{g}E_c^{HH}$. When we detect a vertical edge (when ">"), we refine its direction by checking:

assume ${}^{d}_{q}E_c^{HH}\ if\ |Yc-Yr|>|Yd-Yq|$ (from *q* to *d*)

assume ${}^{d}_{q}E_c^{HH}\ if\ |Yc-Yr|<|Yd-Yq|$ (from *r* to *c*)

assume ${}^{A3}_{A1}E_c^{HH}$ *if* $|Yc - Yr| = |Yd - Yq|$ (strictly vertical)

Else, we assume a homogeneous area. Finally, we compute

$$Y_C^{HD} = \begin{cases} Clip_{32}^R(-3Y_f + 19Y_g + 19Y_h - 3Y_o) & if\ {}^h_g E_c^{HH} \\ Clip_{32}^R(-3Y_c + 19Y_d + 19Y_q - 3Y_r) & if\ {}^d_q E_c^{HH} \\ Clip_{32}^R(-3Y_d + 19Y_c + 19Y_r - 3Y_q) & if\ {}^c_r E_c^{HH} \\ Clip_2^R(Y_g + Y_h) & otherwise \end{cases} \quad (3)$$

Similarly, the generation of the HV pixel begins by examining $|(Y_c + Y_g) - (Y_q + Y_u)| < |(Y_f + Y_p) - (Y_h + Y_r)|$ to detect a vertical edge ${}^g_q E_c^{HV}$. If we detect a horizontal edge (>), we refine its direction and we compute the pixel Y_c^{HV} as follows:

assume ${}^h_p E_c^{HH}$ if $|Y_f - Y_r| > |Y_p - Y_h|$ (from p to h)

assume ${}^r_f E_c^{HH}$ if $|Y_f - Y_r| < |Y_p - Y_h|$ (from f to r)

assume ${}^{A5}_{A4}E_c^{HH}$ if $|Y_f - Y_r| = |Y_p - Y_h|$ (strictly horizontal)

$$Y_C^{HV} = \begin{cases} Clip_{32}^R(-3Y_c + 19Y_g + 19Y_q - 3Y_u) & if\ {}^q_g E_c^{HV} \\ Clip_{32}^R(-3Y_f + 19Y_h + 19Y_p - 3Y_r) & if\ {}^h_p E_c^{HV} \\ Clip_{32}^R(-3Y_h + 19Y_f + 19Y_r - 3Y_p) & if\ {}^r_f E_c^{HV} \\ Clip_2^R(Y_g + Y_q) & otherwise \end{cases} \quad (4)$$

To conclude the CxScale description, we refer to the HD pixel generation, which begins by examining $|(Y_b + Y_g) - (Y_r + Y_w)| > |(Y_e + Y_h) - (Y_q + Y_t)|$ to detect an edge at ${}^h_q E_c^{HD}$. Otherwise, we assume ${}^g_r E_c^{HD}$. Then

$$Y_C^{HD} = \begin{cases} Clip_{32}^R(-3Y_e + 19Y_h + 19Y_q - 3Y_t) & if\ {}^h_q E_c^{HD} \\ Clip_{32}^R(-3Y_b + 19Y_g + 19Y_r - 3Y_w) & if\ {}^g_r E_c^{HD} \end{cases} \quad (5)$$

3. Performance Evaluation

To evaluate the performance of the interpolation techniques in the considered application, we execute multiple motion estimation procedures and the entire application is completed by including the standard H.264/AVC motion compensation. For the realization of each test,

http://dx.doi.org/10.5772/51703

we let the estimation procedure to employ one of the six interpolation techniques described in the previous Section, which will detect the fractional motion. The compensation procedure bases solely on the resulting motion vectors for constructing the frame-predictors according to the standard 6-tap filter. Hence, we use a setup, which ensures that the encoder and the decoder will still be able to use identical reference frames for their predictions, i.e., we avoid the accumulation of errors introduced to the coding process due to the encoder and the decoder. More specifically, the estimation algorithm computes the Sum of Absolute Differences (SAD) for comparing 4×4 pixel candidates and it operates in two phases:

1. A "Diamond Search" matches the block to the best integer position candidate,
2. An exhaustive search in the vicinity of the integer match detects fractional motion by examining 8 candidate blocks located at distance $\pm\frac{1}{2}$ pixels.

Overall, the only parameter varying in this scheme is the interpolation technique used in the second phase of the algorithm, and thus, the quality variations among the output sequences (predictor frames) depend only on the efficiency of the interpolation. The results are shown in the following test reports, which display the PSNR of the output sequences and in particular, the DPSNR for each interpolation technique.

We have performed the simulations to measure the quality and the processing time by testing a variety of well-known videos and up to five (5) frame resolutions for each. The simulations setup with *videos*, number of frames and *resolution* has been: The *car-phone* with 90 frames, the *foreman* with 400 frames, the *container* with 300 frames in *QCIF*, the *coastguard, foreman, news* with 300 frames each in *CIF* and finally, the *blue sky, pedestrian, riverbed, rush-hour* with 100 each in *SD1*, *720p* and *1080p*. Our prediction engine is written in C, it uses 1 reference

Filter:	**H.264**	**Nearest Neighbor**	**Bicubic**			**Lanczos**	**CxScale**	**DDT**
Resolution			**a=-1**	**a=-0.5**	**a=-0.75**			
QCIF	35.0379	-2.2069	-0.0142	-0.0214	-0.0105	0.0009	-0.3359	-0.2265
CIF	34.2930	-1.4229	-0.0150	-0.0340	-0.0166	-0.0042	-0.3697	-0.1994
SD1	33.1775	-0.5483	-0.0170	-0.0192	-0.0118	-0.0030	-0.2071	-0.1249
720p	32.3743	-0.3316	-0.0130	-0.0151	-0.0096	-0.0029	-0.1021	-0.0866
1080p	33.0837	-0.2084	-0.0122	-0.0172	-0.0123	-0.0042	-0.0810	-0.0697
total	**33.4971**	**-0.8843**	**-0.0144**	**-0.0209**	**-0.0120**	**-0.0027**	**-0.2116**	**-0.1372**

Table 1. PSNR of the H.264/AVC filter and DPSNR of other techniques when estimating in HH+HV positions (with H. 264 compensation).

frame and it is designed to efficiently substitute any filter. We begin by distinguishing between horizontal/vertical and diagonal interpolation. Table 1 reports the PSNR results of the algorithm examining fractional displacements only at the horizontal and vertical directions (4 candidates). The table shows the results of two 6-tap filters (H.264/AVC, Lanczos), three

4-tap filters (Bicubic), and two edge-detection based techniques (DDT, CxScale). Moreover, for sake of comparison, we include the PSNR results achieved by the Nearest Neighbor (NN) technique [8]. The table 1 shows the low PSNR results of the Nearest Neighbor (NN) technique [8], which evades interpolation computations by simply forwarding the value of the integer pixel next to the HH/HV position. This technique practically, does not involve fractional motion detection.

The NN results point out that, even with only 4 HH/HV candidates, the algorithm improves its prediction quality by up to 2 dB at low frame resolutions. Using another technique, the Lanczos 6-tap filter, results in almost equivalent quality with the standard H.264 filter. We approximated the Lanczos coefficients by integer values to achieve low complexity operations.

The exact values of the coefficients were set after extensive testing to $\langle 3, -17, 78, 78, -17, 3\rangle$. The performance of the remaining filters lies between the above two extremes of six taps (Lanczos) and zero taps (NN). More precisely, the best quality was achieved with the Bicubic filters. We have examined the performance of several Bicubic kernels, with parameters $-a \in \left\{\frac{7}{8}, \frac{6}{8}, \frac{5}{8}, \frac{4}{8}, \frac{3}{8}, \frac{2}{8}\right\}$ and we report the most prominent of these in Table 1. As it is shown, for most frame resolutions the kernel with coefficients $\langle -3, 19, 19, -3\rangle$ maximizes the quality and limits the expected PSNR degradation to almost 0.01 dB compared to the H.264 filter. That is, although the kernel with coefficients $\langle -1, 5, 5, -1\rangle$ seems –intuitively– a better approximation of the $\langle 1, -5, 20, 20, -5, 1\rangle$ kernel of H.264 (approximation achieved by merging the marginal taps, i.e., by assuming equal values for the corresponding pixels), the experimental results are in favor of a=–0.75. For this reason, CxScale adopts the kernel with coefficients $\langle -3, 19, 19, -3\rangle$ for its Bicubic filtering. Edge-detection based techniques degrade the quality by 0.1 dB, a fact indicating that their induced error surface deviates from the 6-tap filters error surface. However, we note that if we omit the H.264 compensation, these edge-detection based techniques prevail in terms of PSNR, as well as subjective criteria, up to 0.1 dB even when they are compared to 6-taps filters and especially in high-definition videos. Table 1 shows

Filter:	**H.264**	**Nearest Neighbor**	**Bicubic**			**Lanczos**
Resolution			**a=-1**	**a=-0.5**	**a=-0.75**	
QCIF	34.7318	-1.8864	-0.0288	-0.0436	-0.0143	0.0004
CIF	33.9850	-1.1102	-0.0145	-0.0423	-0.0148	-0.0016
SD1	33.1292	-0.4790	-0.0247	-0.0241	-0.0117	-0.0032
720p	32.3766	-0.3178	-0.0176	-0.0188	-0.0092	-0.0031
1080p	33.0869	-0.1979	-0.0146	-0.0223	-0.0119	-0.0045
total	**33.3785**	**-0.7512**	**-0.0202**	**-0.0292**	**-0.0121**	**-0.0004**

Table 2. PSNR of the H.264/AVC filter and DPSNR of Nearest Neighbor, Bicubic and Lanczos when estimating in HD positions (with H.264 compensation).

Filter: / Resolution	CxScale	mDDT	[11]	CrossHD	mDDT′
QCIF	-0.1010	-0.1728	-0.1095	-0.1299	-0.1595
CIF	-0.0740	-0.1731	-0.1219	-0.1414	-0.1595
SD1	-0.0396	-0.1217	-0.0860	-0.0972	-0.1070
720p	-0.0294	-0.0816	-0.0636	-0.0696	-0.0760
1080p	-0.0420	-0.0746	-0.05889	-0.0637	-0.0725
total	**-0.0495**	**-0.1220**	**-0.0864**	**-0.0984**	**-0.1121**

Table 3. DPSNR of CxScale, mDDT, [11], CrossHD and *mDDT′* when estimating in HD positions (with H.264 compensation).

that the performance of the DDT and the CxScale techniques improves as the frame resolution increases.

Next, we consider the report of results regarding the efficiency of the techniques interpolating half-diagonal pixels, which are more computationally demanding than the interpolation of HH/HV pixels. We program the search procedure to examine only 4 HD candidates. Tables 2 and 3 present the resulting PSNR for the techniques of Table 1, plus four edge-detection based techniques: CrossHD, the proposed HD generation based on DDT (mDDT), its first alternative (*mDDT′*), and the technique of [11] using bilinear filtering at its last stage. We can mention here that, when compared to the HH/HV candidates, the HD candidates add slightly less quality to the algorithm, especially in low resolution videos (e.g., as reported in the NN results). Qualitatively, we draw similar conclusions with Table 1 verifying that the Bicubic filtering, especially the kernel with values $\langle$-3,19,19, -3$\rangle$ prevails the over edge-detection based techniques. However, the latter show different behavior when compared to the HH/HV case. More precisely, we deduce that the HD part of CxScale employs an effective

Interpolation technique	PSNR(dB)			Time (μsec)per MB
	QCIF	SD1	1080p	
H.264/AVC	35.4263	33.3687	33.1513	46.0
Lanczos	-0.0032	-0.0050	-0.0076	46.0
Bicubic, a=-0.75	-0.0215	-0.0177	-0.0202	30.6
DDT⊕ mDDT	-0.3513	-0.1782	-0.1018	21.3
DDT⊕ mDDT′	-0.3341	-0.1642	-0.0980	16.0
CxScale	-0.3801	-0.1798	-0.0904	45.6
DDT⊕ CrossHD	-0.3192	-0.1618	-0.0932	32.4
DDT⊕ CxSc(HD)	-0.3061	-0.1302	-0.0839	28.3
DDT⊕ [11](HD)	-0.2913	-0.1492	-0.0889	53.6

Table 4. Quality vs. Time when estimating in HH+HV+HD positions.

gradient check, which is combined with the Bicubic kernel to improve the quality of CxScale. Table 3 shows that it is the prevailing edge-detection based technique among these in the paper. In cases where the filters are using less taps, the CrossHD technique performs better than the DDT techniques.

We complete the evaluation by examining all 8 candidates and taking into account the examination of all pixels at HH, HV, and HD positions. For each technique, Table 4 reports the PSNR results and the time required (as a complexity measure) for generating 16×16 arbitrary half-pixels (averaging over HH, HV, and HD positions) as measured on a Core 2 x86-64 GPP architecture at 3GHz. Furthermore, we combine distinct HH/HV techniques and HD techniques by adopting the prevailing edge-detection mechanisms given in Tables 1, 2 and 3 (in Table 4, "$A \oplus B$" stands for "use technique A in HH/HV interpolation and technique B in HD"). Overall, Bicubic reduces the 6-tap filtering time by 33% and keeps the PSNR level as close as 0.02 dB to the maximum. DDT techniques reduce time by 65% (primarily due to the fast HD generation) with a cost of 0.1 dB. CxScale and [11], involve the time consuming gradient checks. However, the HD part of CxScale combined with DDT (for HH/HV) results in a hybrid technique featuring best PSNR among the edge-detection based techniques with almost 40% time improvement.

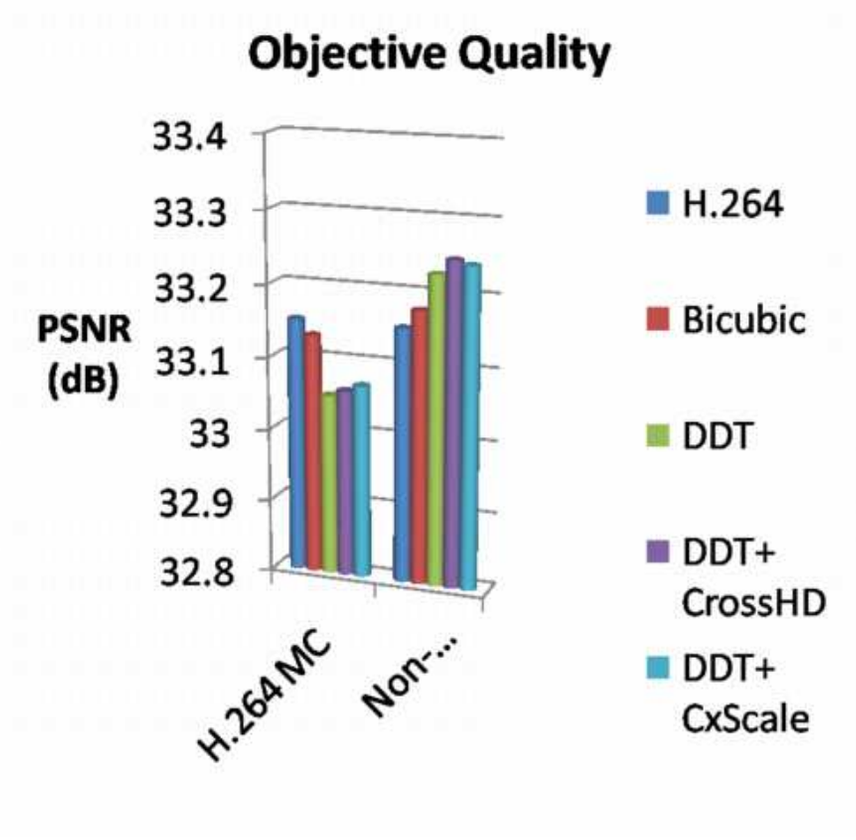

Figure 2. Comparison of objective quality for 5 distinct interpolation procedures. Objective quality is shown both for conventional H.264 and custom motion compensated prediction frames.

http://dx.doi.org/10.5772/51703

In Fig. 2 we show the results of the Objective quality both for conventional H.264 and custom motion compensated prediction frames. Custom motion compensation utilizes the interpolation filter used by the estimation procedure, whereas, conventional compensation uses the H.264 6-tap filter. Several videos of varying resolution were used (QCIF to 1080p). Moreover, Fig. 3 shows how the aforementioned techniques perform with respect to the execution time. Fig.2 shows that best results are achieved by the DDT (in computing HH and HV) with CrossHD (in computing HV). The fastest technique among all presented here, is the DDT with CxScale, which also results in the best PSNR when it is used with the H.264 standard compensation.

Figures 4 show interpolated images of the foreman cif sequence (352x288). We use four distinct interpolation methods at 4x in both directions to subjectively compare the quality of their results. In all four cases, the quarter pixels are calculated with a simple 2-tap bilinear (averaging) filter, which takes as input the two neighboring integer- or half-pixels (computed in a previous iteration by one of the four methods under evaluation).

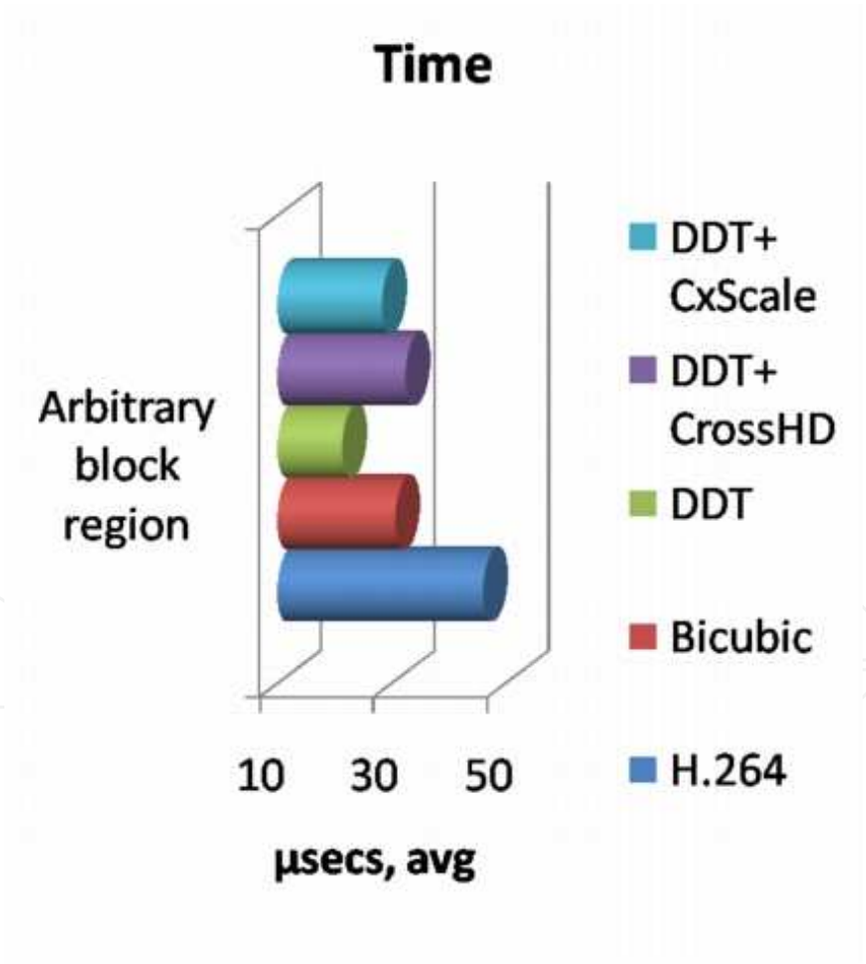

Figure 3. Comparison of execution time for 5 distinct interpolation procedures. Custom motion compensation utilizes the interpolation filter used by the estimation procedure, whereas, conventional compensation uses the H.264 6-tap filter. Several videos of varying resolution were used (QCIF to 1080p).

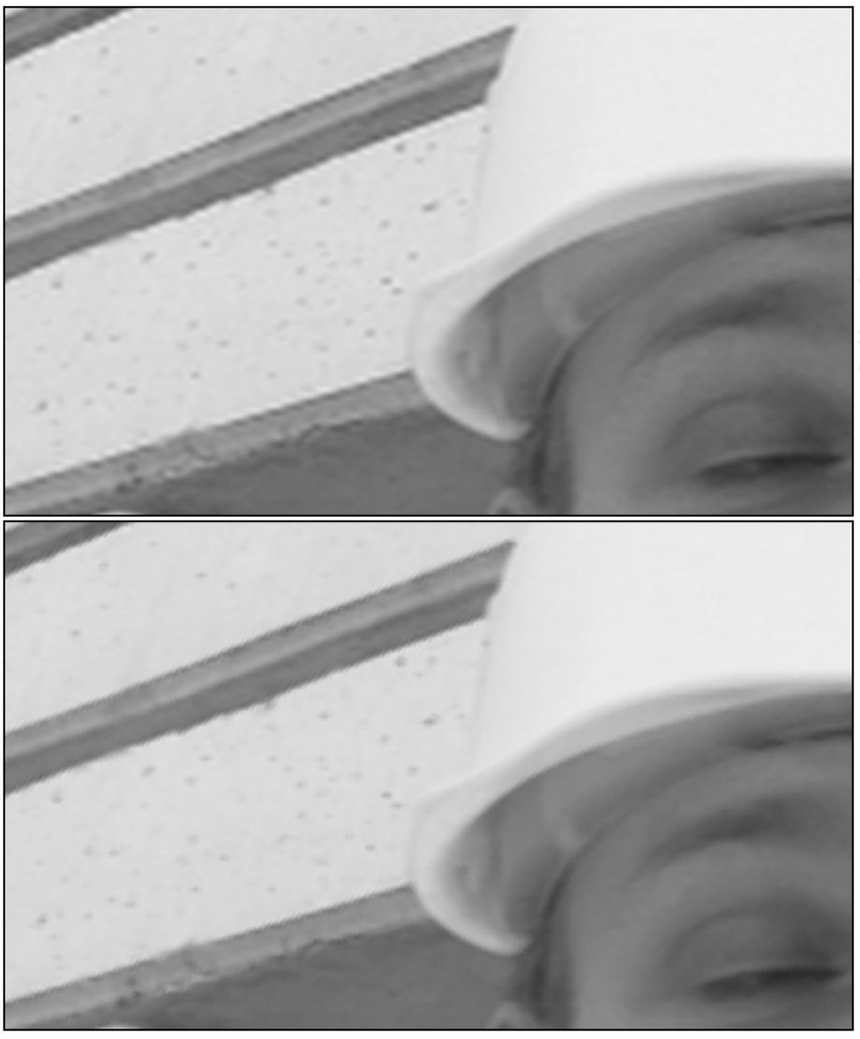

Figure 4.-5. Comparison of the H.264 filter (up) to the DDT ⊕ CxScale (down) on the "foreman" sequence. The example shows the two frames at their increased size (1408x1152) after interpolation from cif (352x288). DDT ⊕ CxScale (down) alleviates aliasing effects.

Figure 6 compares the 6-tap H.264 filter (up) to the combination of DDT and CxScale (down). Clearly, the latter produces much better images in terms of aliasing artifacts: the marquee indents on the wall look much sharper on the image below and the helmet is less jagged. Even though DDT ⊕ CxScale uses less taps, it achieves such aliasing reduction due to the employed edge detection mechanism. However, using a small number of taps and a large area as input to the proposed low-complexity comparison-based mechanism could obscure some finer details. Overall, DDT ⊕ CxScale improves the subjective quality of the enlarged image by using less execution time compared to the examined 6-tap filters. Figures 6 compare the combination of DDT ⊕ CrossHD (up) to the combination of DDT ⊕ [11] (down). Subjectively, the DDT ⊕ CrossHD method uses half the execution time of DDT ⊕ [11] to output images with very similar quality. Both methods reduce the aliasing artifacts compared to the examined 6-tap filters.

http://dx.doi.org/10.5772/51703

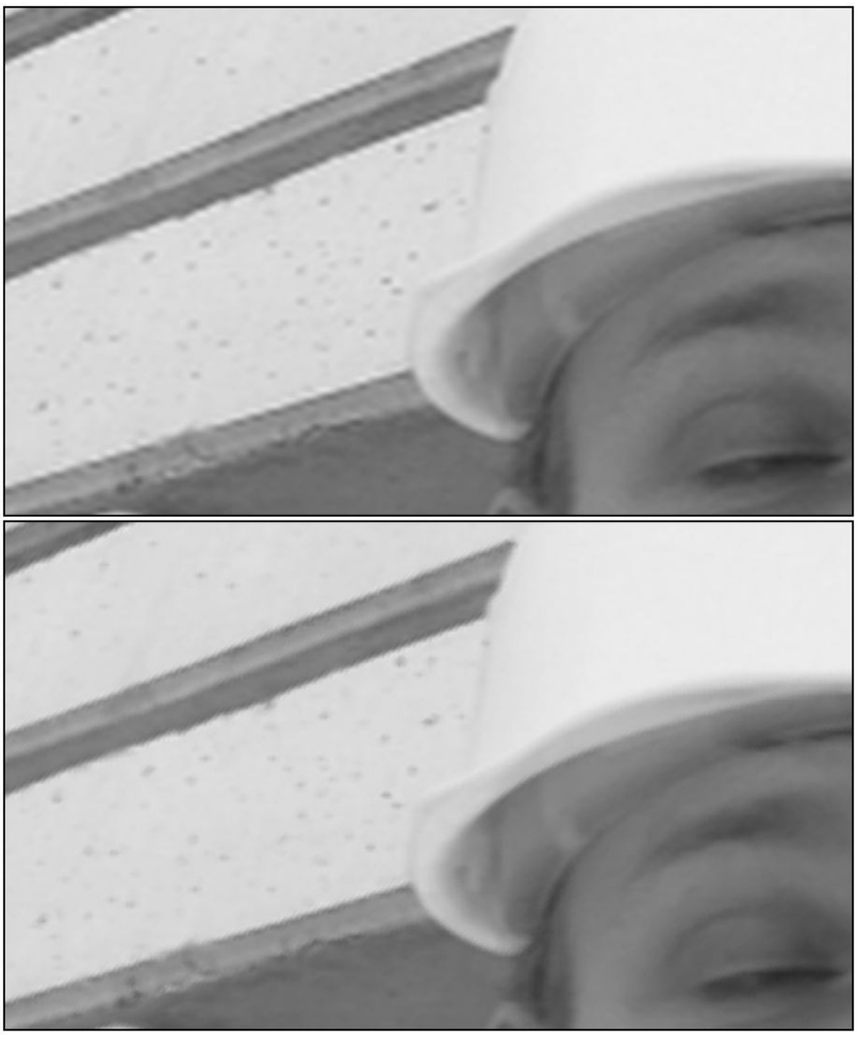

Figure 6.-7. Comparison of the DDT ⊕ CrossHD filter (up) to the DDT ⊕ [11] (down). Frames are shown at their increased size (1408x1152) after interpolation from "foreman" cif (352x288). DDT ⊕ CrossHD produces very similar subjective quality results to DDT ⊕ [11] in considerably less execution time.

4. Conclusion

Aiming at a significant complexity reduction under negligible video quality degradation, the paper proposed three novel interpolation techniques for use in the estimation process preceding the standard H.264/AVC motion compensation module of the encoder. Moreover, we evaluated their performance and compared their efficiency to three commonly used techniques. The results showed that the techniques using 4-tap Bicubic kernels constitute the most prominent substitute of the standard 6-tap filter. Further reduction of the estimation time was achieved via combinations of simple edge-detection based techniques. Future work includes parallelized implementations in VLSI/FPGA and cost-performance analysis.

Author details

Georgios Georgis, George Lentaris and Dionysios Reisis*

*Address all correspondence to: dreisis@phys.uoa.gr

Electronics Laboratory, Physics Deparment, National and Kapodistrian University of Athens (NKUA), Greece

References

[1] Yu-Wen, Huang, Bing-Yu, Hsieh, Shao-Yi, Chien, Shyh-Yih, Ma, & Liang-Gee, Chen. (2006). Analysis and Complexity Reduction of Multiple Reference Frames Motion Estimation in H.264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology*, doi :10.1109/TCSVT.2006.872783, 16(4), 507-522.

[2] Tung-Chien, Chen, Yu-Wen, Huang, & Liang-Gee, Chen. (2004). Fully utilized and reusable architecture for fractional motion estimation of H.264/AVC. *IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, doi : 10.1109/ICASSP. 2004.1327034, 9-12.

[3] ITU, Telecommunication Standardization Sector. (2010). Advanced Video Coding for Generic Audiovisual Services. *ITU-T*, 167-169, Mar.

[4] Gupta, P. S. S. B. K., & Korada, R. (2004). Novel algorithm to reduce the complexity of quarter-pixel motion estimation. *Proc. of Visual Communications and Image Processing*, 5308, 31-36, Jan, doi: 10.1117/12.532336.

[5] Hyun, C. J., & Sunwoo, M. H. (2009). Low Power Complexity-Reduced ME and Interpolation Algorithms for H.264/AVC. *J. of Signal Processing Systems*, 56(2), 285-293, Sept, doi : 10.1007/s11265-008-0224-4.

[6] Changqi, Yang, Goto, S., & Ikenaga, T. (2006). High performance VLSI architecture of fractional motion estimation in H.264 for HDTV. *IEEE Intl. Symposium on Circuits and Systems (ISCAS)*, September, doi : 10.1109/ISCAS.2006.1693157.

[7] Chao-Yang , Kao, Cheng-Long, Wu, & Lin, Youn-Long. (2010). A high performance three-engine architecture for H.264/AVC fractional motion estimation. *IEEE Tran. On Very Large Scale Integration Systems*, April, doi : 10.1109/ICME.2008.4607389, 18(4), 662-666.

[8] Dodgson, N. A. (1997). Quadratic Interpolation for Image Resampling. *IEEE Trans. on Image Processing*, 6(9), 1322-1326, Sept, doi: 10.1109/83.623195.

[9] Keys, R.G. (1981). Cubic Convolution Interpolation for Digital Image Processing. *IEEE Transactions on Acoustics, Speech and Signal Processing*, Dec, doi : 10.1109/TASSP. 1981.1163711, 29(6), 1153-1160.

[10] Burger, W., & Burge, M. (2008). Digital Image Processing, an Algorithmic approach using Java. *1st ed. New York, USA: Springer*.

[11] Su, D., & Willis, P. (2004). Image Interpolation by Pixel-Level Data-Dependent Triangulation. *Computer Graph. For.*, doi : 10.1111/j.1467-8659.2004.00752.x, 23(2), 189-201.

[12] Chen, Tung-Chien, Huang, Yu-Wen, & Chen, Liang-Gee. (2004). Analysis and design of macroblock pipelining for H.264/AVC VLSI architecture. *IEEE Intl. Symp. on Circuits and Systems (ISCAS)*, 273-276, doi : 10.1109/ISCAS.2004.1329261.

[13] Hyun, C. J., Kim, S. D., & Sunwoo, M. H. (2006). Efficient memory reuse and sub-pixel interpolation algorithms for ME/MC of H.264/AVC. *IEEE Workshop on Signal Processing Systems Design and Implementation*, October, doi : 10.1109/SIPS.2006.352612, 377-38.

[14] Song, Y., Ma, Y., Liu, Z., Ikenaga, T., & Goto, S. (2008). Hardware-oriented direction-based fast fractional motion estimation algorithm in H.264/AVC. *IEEE International Conference on Multimedia and Expo*, 1009-1012, June, doi : 10.1109/ICME.2008.4607608.

[15] Vatis, Y, & Ostermann, J. (2009). Adaptive Interpolation Filter for H.264/AVC. *Circuits and Systems for Video Technology, IEEE Transactions on*, 19(2), 179-192, Feb., doi: 10.1109/TCSVT.2008.2009259.

[16] Hsueh-Ming, Hang, Peng, Wen-Hsiao, Chia-Hsin, Chan, & Chun-Chi, Chen. (2010). Towards the Next Video Standard: High Efficiency Video Coding. *Proceedings of the Second APSIPA Annual Summit and Conference*, 609-618, Biopolis, Singapore, 14-17 December.

[17] Dmytro, Rusanovskyy, Ugur, Kemal, Hallapuro, Antti, Lainema, Jani, & Gabbouj, Moncef. (2009). Video Coding With Low-Complexity Directional Adaptive Interpolation Filters. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(8), August, doi : 10.1109/TCSVT.2009.2022708.

[18] Fuldseth, A., Bjontegaard, G., Rusanovskyy, D., Ugur, K., & Lainema, J. (2008). Low complexity directional interpolation filter. Berlin, Germany, ITU-T Q.6/SG16, VCEG-AI12, July.

[19] Zhang, Kai, Guo, Xun, An, Jicheng, Huang, Yu-Wen, Lei, S., & Gao, Wen. (2012). A Single-Pass-Based Localized Adaptive Interpolation Filter for Video Coding. *Circuits and Systems for Video Technology, IEEE Transactions on*, 22(1), 43-55, Jan., doi: 10.1109/TCSVT.2011.2157194.

[20] Cho, Jaehyun, Lee, Dong-Bok, Cheol, Shin Jeong, & Song, Byung Cheol. (2011). Block-adaptive interpolation filter for sub-pixel motion compensation. *19th European Signal Processing Conference (EUSIPCO)*, 2156-2160.

[21] Georgis, G, Lentaris, G, & Reisis, D. (2012). Study of Interpolation Filters for Motion Estimation with Application in H.264/AVC Encoders. *IEEE Intl. Conference on Circuits and Systems (ICECS)*, 9-12, Beirut, doi : 10.1109/ICECS.2011.6122201, 9-12.

IntechOpen