## Quarter-Pixel Accuracy Motion Estimation (ME) - A Novel ME Technique in HEVC

Kalyan Kumar Barik<sup>1,\*</sup>, Somnath Sengupta<sup>1</sup>, Manas Ranjan Jena<sup>2</sup>

<sup>1</sup>Department of E&EC, IIT, KHARAGPUR <sup>2</sup>Department of ETC, Department of ELTCE, VSSUT, BURLA, ODISHA \*Corresponding author: kalyankumar333@gmail.com

Received May 30, 2014; Revised June 19, 2014; Accepted June 19, 2014

**Abstract** In this paper, We have focused on the development and implementation of novel low computational cost motion estimation(ME) algorithm for video coding based on H.265/HEVC standard. Through this algorithm, new ideas are explored for potentially improving the standard. High-Efficiency Video Coding (HEVC) is a new video compression standard currently being standardized by the JCTVC (Joint Collaborative Team on Video Coding) established by ISO/IEO MPEG and ITU-T. HEVC targets 50% coding gain over AVC/H.264 High Profile. In order to achieve this goal, new tools are being adopted to HEVC. One of the main differences of HEVC from previous video compression standards is the coding tree structure. In this structure, a frame is first divided into CTUs (coding tree units). Then a CTU is further divided into CUs (coding units) in a quad-tree structure. Unless a CU is further divided into four smaller CUs, it is predicted with one of several PU (prediction unit) types. Currently, CTU size can be as large as 64x64 and SCU (smallest CU) can be as small as 8x8. At the encoder side, decisions can be made to support only a subset of these PU types and CU sizes. This will present a trade-off between hardware complexity and coding efficiency. Since motion estimation is one of the most critical blocks in video encoders, trade-offs in motion estimation (ME) block are very important for the overall encoder design. However, it is important to consider the memory cost (in terms of area and data bandwidth) of these tools in hardware. This paper presents HEVC ME using Quarter -pixel accuracy prediction using 8-tap and 7- tap filter for luma interpolation using 2D Logarithmic search and goal is to reduce number of motion vector (MC) to increase compression gain, PSNR and to reduce memory area and data band data bit rate.

Keywords: ME, HEVC, AVC, MC, MV, coding block, coding unit, SAO

**Cite This Article:** Kalyan Kumar Barik, Somnath Sengupta, and Manas Ranjan Jena, "Quarter-Pixel Accuracy Motion Estimation (ME) - A Novel ME Technique in HEVC." *International Transaction of Electrical and Computer Engineers System*, vol. 2, no. 3 (2014): 107-113. doi: 10.12691/iteces-2-3-5.

### 1. Introduction

Recent advances in digital technologies have paved the way to the development of numerous real-time applications deemed too complex in the past. Digital video has been a regular presence in our lives for many years now. Whether used for digital television, in personal computers, hand held devices or other multimedia applications, its use has grown tremendously in the last years and it seems that this growth is not slowing down. A vast array of those applications requires transmission and storage of digital videos. Examples include but are not limited to: digital TV, video streaming, multimedia communications, remote monitoring, videophones and video conferencing. Advances in digital video can be classified as one of the most influential modern technologies; this is due to the fast wide spread use of digital video applications into everyday life. Consequently, over the last three decades, high-quality digital video has been the goal of companies, researchers and standardisation bodies [1,9].

The main idea behind video compression is to remove redundancies from the signals. This is carried out in the spatial domain within individual frames and in the temporal domain between neighbouring frames. Due to the fact video frames are typically displayed at a frame rate of 20 to 33 frames per second to the user, it is easy to understand that neighbouring frames often show high resemblance, hence removing the temporal redundancy can accomplish high compression ratios in practice. This is normally achieved in two steps: first, a Motion Estimation (ME) technique is used to calculate the motion distance, where blocks are defined areas within the frame. Secondly, with the available motion information, the residual between the current encoded frame and the previous frame is compensated and what is called Motion Compensation (MC) [9].

Motion estimation (ME) and motion compensation is seen as one of the most important methods of exploiting redundancy in motion pictures. Its importance is so high that 50% to 70% of encoder complexity is dedicated to the motion estimation process. However, as we move towards higher resolution videos, computational complexity is becoming a bigger concern [2]. This is why motion estimation is seen as a major savings area in terms of computational expense. An important technique that can be used in ME is the Fractional Motion Estimation (FME). The FME allows an even greater efficiency by applying an interpolation process between integer position samples in reference frames, allowing a search for better matches in fractional positions. The FME is composed by two units: the Interpolation Unit, that generates the fractional position samples (sub-pixels), and the Search Unit, which searches for better matches composed by sub-pixels. One of the new FME is "Quarter-pixel accuracy motion estimation", which is used for the most efficient video coding standards H.264/MPEG-4(AVC) and H.265/HEVC [4,5].

### 2. Motion Estimation in HEVC

High-Efficiency Video Coding (HEVC) is a new video compression standard currently being standardized by the JCTVC (Joint Collaborative Team on Video Coding) established by ISO/IEO MPEG and ITU-T. HEVC targets 50% coding gain over AVC/H.264 High Profile [2]. In order to achieve this goal, new tools are being adopted to HEVC. The main goals of H.265/HEVC standardization effort have been to enhance compression performance and provide a "network-friendly" video representation. H.265 utilizes variable block sizes and quarter-pixel motion compensation with multiple reference frames to achieve high coding efficiency. It has motion compensation units in sizes of 8x8, 16x16, 32x32 and 64x64 where H.264/AVC supports only 4x4 to 16x16 [6]. Such wide block choices improve coding efficiency at the cost of largely increased motion estimation time [3]. In H.265/HEVC encoding, the most computationally critical part is motion estimation [9].

H.265/HEVC also has quarter-pixel motion vector accuracy as another of its important feature, which requires interpolation of pictures by a factor of four, which is done by a 7-tap bilinear filter and a 8-tap DCT (Discrete Cosine Transform) based finite impulse response (FIR) filter [2]. This increased accuracy of motion vectors and the subsequent coding gain is significant. On the other hand, the filtering process and the extra quarter-pixel motion estimation search demands substantial amount of computation. The computational complexity becomes even worse with larger search ranges, bi-directional and/or when multiple reference frames are used. Such high computational complexity is often a bottle-neck for realtime conversational applications.

# **2.1. Sub-Pixel Interpolation Based Motion** Estimation

Sub-pixel interpolation is one of the most computationally intensive parts of High Efficiency Video Coding (HEVC) video encoder and decoder. The fractional sample interpolation for luma samples in HEVC [3] uses separable application of an eight-tap filter for the half-sample positions and a seven-tap filter for the quarter sample positions. This is in contrast to the process used in H.264/MPEG-4 AVC [5], which applies a two-stage interpolation process by first generating the values of one or two neighbouring samples at half-sample positions using six-tap filtering, rounding the intermediate results, and then averaging two values at integer or half- sample positions. In H.264, 4x4 and 16x16 block sizes are used. However, in HEVC, prediction unit (PU) sizes can be from 4x4 to 64x64.Therefore, HEVC sub-pixel interpolation is more complex than H.264 sub-pixel interpolation [8].

HEVC instead uses a single consistent separable interpolation process for generating all fractional positions without intermediate rounding operations, which improves precision and simplifies the architecture of the fractional sample interpolation. The interpolation precision is also improved in HEVC by using longer filters, i.e., seven-tap or eight-tap filtering rather than the six tap filtering used in H.264/MPEG-4 AVC [2,10,11]. Using only seven taps rather than the eight used for half-sample positions was sufficient for the quarter-sample interpolation positions since the quarter-sample positions are relatively close to integer sample positions, so the most distant sample in an eight-tap interpolator would effectively be farther away than in the half sample case (where the relative distances of the integer-sample positions are symmetric).

A.11	A <sub>0,-1</sub>	a <sub>0,-1</sub>	b <sub>0,-1</sub>	с <sub>0,-1</sub>	A <sub>1,-1</sub>	A <sub>2,-1</sub>
			2			
A-1,0	A <sub>0,0</sub>	a <sub>0,0</sub>	b <sub>0,0</sub>	C <sub>0,0</sub>	A <sub>1,0</sub>	A <sub>2,0</sub>
d_1,0	d <sub>0,0</sub>	e <sub>0,0</sub>	f <sub>0,0</sub>	B0,0	d <sub>1,0</sub>	d <sub>2,0</sub>
h-1,0	h <sub>0,0</sub>	i <sub>0,0</sub>	jo,o	k <sub>0,0</sub>	h <sub>1,0</sub>	h <sub>2,0</sub>
n. <sub>1,0</sub>	n <sub>0,0</sub>	P0,0	<b>q</b> <sub>0,0</sub>	r <sub>0,0</sub>	n <sub>1,0</sub>	n <sub>2,0</sub>
A.1,1	A <sub>0,1</sub>	a <sub>0,1</sub>	b <sub>0,1</sub>	C <sub>0,1</sub>	A <sub>1.1</sub>	A <sub>2,1</sub>
A.1.1	A <sub>0,1</sub>	a <sub>0,2</sub>	b <sub>0,2</sub>	C <sub>0,2</sub>	A <sub>1.1</sub>	A <sub>2,1</sub>

Figure 1. Integer and fractional sample positions for luma interpolation

#### 2.2. Interpolation Process of Luma Sample

In Figure 2.3 the positions labeled with upper-case letters Ai, j, represent the available luma samples at integer sample locations, whereas the other positions labelled with lower-case letters represent samples at non integer sample locations, which need to be generated by interpolation. The samples labelled  $\mathbf{a}_{0,0}$ ,  $\mathbf{b}_{0,0}$ ,  $\mathbf{c}_{0,0}$ ,  $\mathbf{h}_{0,0}$ ,  $\mathbf{and}$   $\mathbf{n}_{0,0}$  and are derived from the samples by applying the eight-tap filter for half-sample positions [5] and the seven-tap filter for the quarter-sample positions as follows:

$$a_{0,j} = \left(\sum_{i=-3..3} A_{i,j} q filter[i]\right) >> (B-8)$$
 (2.1)

$$b_{0,j} = \left(\sum_{i=-3..4} A_{i,j} \text{hfilter}[i]\right) >> (B-8)$$
 (2.2)

$$c_{0,j} = \left(\sum_{i=-2..4} A_{i,j} q filter [1-i]\right) >> (B-8)$$
 (2.3)

$$d_{0,0} = \left(\sum_{i=-3..3} A_{0,i} q \text{ filter}[j]\right) >> (B-8) \qquad (2.4)$$

$$h_{0,0} = \left( \sum_{i=-3..4} A_{0,j} \text{hfilter}[j] \right) >> (B-8) \quad (2.5)$$

$$\mathbf{n}_{0,0} = \left(\sum_{i=-2..4} \mathbf{A}_{0,i} q \text{filter}[1-i]\right) >> (B-8) \quad (2.6)$$

where the constant  $B \ge 8$  is the bit depth of the reference samples (and typically B = 8 for most applications) and the filter coefficient values for luma is given in Table 1 [4,5]. In these formulae >> denotes an arithmetic right shift operation.

 Table 1. Filter coefficients for luma fractional sample interpolation in HEVC

Index i	-3	-2	-1	0	1	2	3	4
hfilter[i]	-1	4	-11	40	40	-11	4	1
qfilter[i]	-1	4	-10	58	17	-5	1	

The samples labelled  $\mathbf{e}_{0,0}$ ,  $\mathbf{i}_{0,0}$ ,  $\mathbf{p}_{0,0}$ ,  $\mathbf{j}_{0,0}$ ,  $\mathbf{q}_{0,0}$ ,  $\mathbf{g}_{0,0}$ ,  $\mathbf{k}_{0,0}$ and  $\mathbf{r}_{0,0}$  can be derived by applying the corresponding filters to samples located at vertically adjacent  $\mathbf{a}_{0,j}$ ,  $\mathbf{b}_{0,j}$  and  $\mathbf{c}_{0,j}$  positions as follows:

$$e_{0,0} = \left(\sum_{v=-3.3} a_{0,v} q filter[v]\right) >> 6$$
 (2.7)

$$f_{0,0} = \left(\sum_{v=-3..3} b_{0,v} q filter[v]\right) >> 6$$
 (2.8)

$$g_{0,0} = \left(\sum_{v=-3.3} c_{0,v} q \text{filter}[v]\right) >> 6$$
 (2.9)

$$i_{0,0} = \left(\sum_{v=-3..4} a_{0,v} \text{hfilter}[v]\right) >> 6$$
 (2.10)

$$j_{0,0} = \left(\sum_{v=-3..4} b_{0,v} q \text{ filter}[v]\right) >> 6$$
 (2.11)

$$k_{0,0} = \left(\sum_{v=-3..4} c_{0,v} q filter[v]\right) >> 6$$
 (2.12)

$$\mathbf{p}_{0,0} = \left(\sum_{\mathbf{v}=-2..4} a_{0,\mathbf{v}} q \text{filter}[1-\mathbf{v}]\right) >> 6 \qquad (2.13)$$

$$q_{0,0} = \left(\sum_{v=-2..4} b_{0,v} q filter [1-v]\right) >> 6 \qquad (2.14)$$

$$\mathbf{r}_{0,0} = \left(\sum_{v=-2..4} c_{0,v} q \text{filter}[1-v]\right) >> 6 \qquad (2.15)$$

## **2.3. Interpolation Process of Chrominance Sample**

Figure 2 shows the positions of the integer pixel sample, 1/2 pixel sample, 1/4 pixel sample, 1/8 pixel sample of the chrominance components of the reference image. It is supposed that chrominance sample point  $B_{i, j}$  is located at the integer sample point (xB <sub>i,j</sub>, yB<sub>i, j</sub>), then the predicted value from chrominance point 'ab<sub>0,0</sub>' to 'hh<sub>0,0</sub>' at non-integer sample positions can be obtained by the 4-beat filter with the coefficientient is given in Table 2 [5].

Table 2. Filter coefficients for chroma sample interpolation in HEVC

Index	-1	0	1	2
filter1[i]	-2	58	10	-2
filter2[i]	-4	54	16	-2
filter3[i]	-6	46	28	-4
filter4[i]	-4	36	36	-4

The values of 1/2 pixel points  $ae_{0,0}$ ,  $ea_{0,0}$ ; 1/4 pixel point  $ac_{0,0}$ ,  $ag_{0,0}$ ,  $ca_{0,0}$ ,  $ga_{0,0}$ ; and 1/8 pixel point  $ab_{0,0}$ ,  $ad_{0,0}$ ,  $af_{0,0}$ ,  $ah_{0,0}$ ,  $ba_{0,0}$ ,  $da_{0,0}$ ,  $fa_{0,0}$ ,  $ha_{0,0}$  can be obtained by using filter interpolation mentioned in the Table 2 on the nearest integer pixel in the horizontal and vertical directions and similarly the value of sub-pixel sample point  $bX_{0,0}$ ,  $cX_{0,0}$ ,  $dX_{0,0}$ ,  $eX_{0,0}$ ,  $fX_{0,0}$ ,  $gX_{0,0}$  and  $hX_{0,0}$  (among which, X presents any one in b, c, d, e, f, g and h) can be obtained by the 4-beat filter interpolation in the vertical direction.

	ha <sub>0,-1</sub>	hb <sub>0,-1</sub>	hc <sub>0,-1</sub>	hd <sub>0,-1</sub>	he <sub>0,-1</sub>	hf <sub>0,-1</sub>	hg <sub>0,-1</sub>	hh <sub>0,-1</sub>	
ah <sub>-1,0</sub>	B <sub>0,0</sub>	ab <sub>0,0</sub>	ac <sub>0,0</sub>	ad <sub>0,0</sub>	ae <sub>0,0</sub>	af <sub>0,0</sub>	ag <sub>0,0</sub>	ah <sub>0,0</sub>	B <sub>1,0</sub>
bh <sub>-1,0</sub>	ba <sub>0,0</sub>	bb <sub>0,0</sub>	bc <sub>0,0</sub>	bd <sub>0,0</sub>	be <sub>0,0</sub>	bf <sub>0,0</sub>	bg <sub>0,0</sub>	bh <sub>0,0</sub>	ba <sub>1,0</sub>
ch <sub>-1,0</sub>	ca <sub>0,0</sub>	cb <sub>0,0</sub>	cc <sub>0,0</sub>	cd <sub>0,0</sub>	ce <sub>0,0</sub>	cf <sub>0,0</sub>	cg <sub>0,0</sub>	ch <sub>0,0</sub>	ca <sub>1,0</sub>
dh <sub>-1,0</sub>	da <sub>0,0</sub>	db <sub>0,0</sub>	dc <sub>0,0</sub>	dd <sub>0,0</sub>	de <sub>0,0</sub>	df <sub>0,0</sub>	dg <sub>0,0</sub>	dh <sub>0,0</sub>	da <sub>1,0</sub>
eh_1,0	ea <sub>0,0</sub>	eb <sub>0,0</sub>	ec <sub>0,0</sub>	ed <sub>0,0</sub>	ee <sub>0,0</sub>	ef <sub>0,0</sub>	eg <sub>0,0</sub>	eh <sub>0,0</sub>	ea <sub>1,0</sub>
fh <sub>-1,0</sub>	fa <sub>0,0</sub>	fb <sub>0,0</sub>	fc <sub>0,0</sub>	fd <sub>0,0</sub>	fe <sub>0,0</sub>	ff <sub>0,0</sub>	fg <sub>0,0</sub>	$\mathrm{fh}_{0,0}$	fa <sub>1,0</sub>
gh <sub>-1,0</sub>	ga <sub>0,0</sub>	gb <sub>0,0</sub>	gc <sub>0,0</sub>	gd <sub>0,0</sub>	ge <sub>0,0</sub>	gf <sub>0,0</sub>	<b>gg</b> 0,0	gh <sub>0,0</sub>	ga <sub>1,0</sub>
hh <sub>-1,0</sub>	ha <sub>0,0</sub>	hb <sub>0,0</sub>	hc <sub>0,0</sub>	hd <sub>0,0</sub>	he <sub>0,0</sub>	hf <sub>0,0</sub>	hg <sub>0,0</sub>	hh <sub>0,0</sub>	ha <sub>1,0</sub>
	B <sub>0,1</sub>	ab <sub>0,1</sub>	ac <sub>0,1</sub>	ad <sub>0,1</sub>	ae <sub>0,1</sub>	af <sub>0,1</sub>	ag <sub>0,1</sub>	ah <sub>0,1</sub>	B <sub>1,1</sub>

Figure 2. Positions of Integer Sample Point and Non-integer Sample Point in the Interpolation of Chrominance

### **3. Simulation Results & Analysis**

3.1. Results of Quarter-pixel Motion Estimation

To illustrate the implementation result of quarter-pixel motion estimation in HEVC, experiment have been carried out using MATLAB. Motion vectors are obtained by using simple 2D Logarithmic search algorithm. The characteristics of the motion activities of the blocks in the current frame are predicted using this temporal information. As discussed, we implemented the Quarterpixel interpolation algorithm with implementation of 2D Logarithmic search to find the motion vector, predicted frame with PSNR and residual with motion compensation and the corresponding 3D mesh plot of residual.

The experiment has been carried out by taking different CTU size i.e. 8X8, 16X16, 32X32, and 64X64 of different video frames such are AVI, DIVX and YUV. For 8x8,



**Reference Frame Candidate** 

16X16, 32X32, and 64X64 CTU size the search block size 10x10, 20x20, 40x40 and 70x70 respectively. ME results of only 8x8 CTU of AVI, 16x16 CTU of DIVX and 32x32 CTU of YUV video frames are shown in Figure 3, Figure 4, and Figure 5.

## **3.2. Results of ME by Taking 8x8 CTU of an AVI Video Frame**

In order to find the motion vector, predicted frame with PSNR and with motion compensation and the corresponding 3D mesh plot of residual of an AVI video frames, here the CTU size is considered as 8x8 and the searching block size around CTU is 10x10.



Candidate Frame ,PSNR 27.2146dB



Motion vector field between reference & candidate frame. 8X8 CTU



Predicted Frame with PSNR 33.0892dB, Abs. Difference with MC and 3D Mesh plot

# **3.3. Results of ME by Taking 16x16 CTU of a DivX Video Frame**

In order to find the motion vector, predicted frame with PSNR and with motion compensation and the corresponding 3D mesh plot of residual of a DIVX video frames, here the CTU size is considered as 16x16 and the searching block size around CTU is 20X20.



**Reference Frame Candidate** 



Candidate Frame ,PSNR 27.7167dB

-					-																									-										-	_
								-																		-												100	1.00		
-							-		-				1					-	-	-	-	-		-	-	-	-	-					-					-	-	-	-
										-	-	-	-	-	-	-	-		-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-	-	-	-	-	
-		-				-			-	-	-	-	-	-	-			-	-	-	-	-	-	-		-	100	-	-	100	-		-	-	1						-
		-				-		-	-	-	-	-	-	-	-		-		-	-	-	-	-	-			-	-	-				-		-		-	-			
-		-				-		-	-	-	-	-	-	-	-	-		-	-	-	-	-	1	-		-	-	-					-	-		-					-
		-		-				-	-	-	-	-	-	-	-	-		-	-	-	-	-	~	-	1.00	-	-	-	-	100	100	-		-	-	-	-		-		
-			-	-	-	1	-	-	-	-	-	-	~	-	-	-		1	-	-	-	-	~	~		-								-	-	-			-		-
		-						-	-	-	-	-		-	100	-	-		-	-	-	-	-	-									-	-		1					
-					-	1.00		-	-	-	-	-	-	-		-	-		1	-	-	1	1			-							-				-		-		-
	10	-		-	-	1.00		-		-	-	-	-		-	1.00		-	1	-	1		100	-		-								10							
-			-						-	-	-	- 10	-	-		-		-	-	*			1		-	-								-	1						-
				-						-	-	-	1.00	1				-	-	-		-			-	-	100							-	-		-				
-					-	-			1.0						-					-	-	-			1.00	-		-					-	-		-		-		1.00	-
					-							-					-		-	-	-		-	-		-	-	-	-												
- 21-					-																	-		-	-	-	-	-										-			
																								-	-	-	-											-			
-					+				-	-2	-				-	_			-	-2	_				-24	_			-	*	-			-	ź	-			-	-	

## Motion vector field between reference & candidate frame, 16x16 CTU



## Predicted Frame with PSNR 32.4501dB, Abs. Difference with MC and 3D Mesh plot

Figure 4. Results of ME by taking 16x16 CTU of DIVX video Frames

**3.4. Results of ME by Taking 32x32 CTU of a YUV Video Frame** 

In order to find the motion vector, predicted frame with PSNR and with motion compensation and the corresponding 3D mesh plot of residual of a YUV video frames, here the CTU size is considered as 32x32 and the searching block size around CTU is 40X40.

20



**Reference Frame Candidate** 



Candidate Frame ,PSNR 28.7961 dB

+			-							- +
-					-					- +
-					-	-	-			-
+ -	-				-	-	-		-	-
+ -	-		-		-	- 5	6	-		- +
+					1	- 8	1	1		- +
-		-	-	-	-			1	=	-
-				-	-	-	-	7.		-
-					-		-		-	

Motion vector field between reference & candidate frame, 32x32 CTU



Predicted Frame with PSNR 29.1448dB, Abs. Difference with MC and 3D Mesh plot

Figure 5. Results of ME by taking 32x32 CTU of YUV video Frames

### 4. Result Analysis

The H.265 encoding method has been complicated by the development of new coding tools. Among those tools, the quarter pixel accuracy motion estimation and compensation enhance compression gain and to reduce memory area and data bit rate and it requires the implementation of complex interpolation filters and increases the ME complexity. Here the scheme was divided into four steps; in the first step the sub-pixel ME for the  $8\times8$  and  $16\times16$ , 32x32 and 64x64 block has been used.

The result shows that with increase of size of CTU block, the PSNR of predicted frame gradually decreases

and also the number of motion vector reduces as given in Table 3. The PSNR of original candidate frame was 27.2146dB, 27.7167dB and 28.7961 dB with respect to reference frame for AVI, DIVX and YUV video frames respectively.

Table 3. PSNR of different Vide	eo Frames with different size CTU
---------------------------------	-----------------------------------

А	VI	DI	VX	YUV				
CTU size	PSNR(dB)	CTU size	PSNR(dB)	CTU size PSNR(dB				
8X8	33.0892	8X8	34.2917	8X8	30.8412			
16X16	31.6316	16X16	32.4501	16X16	29.9705			
32X32	29.8129	32X32	30.1317	32X32	29.1448			
64X64	28.364	64X64	29.2345	64X64	28.4357			

### 5. Comparison with H.264/AVC

Quarter-pixel interpolation using 7-tap and 8-tap filter in HEVC gives more details in comparison to 6-tap filter base quarter-pixel interpolation in H.264/AVC. The computational cost of this process can be reduced utilizing parallel processing technique in the hardware implementation. By comparing the PSNR, obviously the PSNR of predicted frame produced by the use of H.264 Quarter-pixel interpolation filter is much less in comparison to Quarter-pixel Interpolation filter of HEVC. The performance of the quarter-pel filters in H.264/AVC is relatively poor, especially the filters for the quarter-pel pixels e, g, p and r in the diagonal direction. In general performance gain (more than 10%) of interpolation filters in HEVC compared to H.264/AVC comes from the quarter-pel interpolations.

### 6. Conclusion

Quarter-pixel interpolation based motion estimation is an optimized process and normally used to increase the compression gain in HEVC and which is here implemented in MATLAB. According to the experimental results, this implementation of quarter-pixel interpolation based motion estimation working as like as HEVC reference software HM 5.2 with the promotion of the next generation video coding standard HEVC. To enable a parallel processing, the macro blocks (CTU) are processed on dedicated and special processor, so the all motion vectors will be outputted for an individual frame at the same time. This technique may increases the requirement of hardware resources but definitely reduces the time and computation complexity and also increases compression gain of encoder.

### References

- Jens-Rainer Ohm and G. J. Sullivan, "High Efficiency Video Coding : The Next Frontier in Video Compression," IEEE Signal Processing Magazine, pp.153-158, January 2013.
- [2] Wang Gang, C. Hexin, C. Maianshu, "A Study on Sub-pixel Interpolation Filtering Algorithim and Hardware Structural Design Aiming at HEVC," Telkomnia, Vol.11, No. 12, pp. 7564-7570, Dec. 2013.
- [3] B. Bross, W. J. Han, J. R. Ohm, G. J. Sullivan, T. Wiegand, "High Efficiency Video Coding (HEVC) Text Specification Draft 7", JCTVC-I1003, May 2012.
- [4] M.T. Pourazad, C. Doutre, M. Azimi, P. Nasiopoulos, "HEVC: The New Gold Standard for Video Compression", IEEE Consumer Electronics Magazine, July 2012.
- [5] G. J. Sullivan, J.-R.Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard", IEEE Trans. Circuits and Systems for Video Technology, Vol. 22, No. 12, pp. 1649-1668, Dec. 2012.
- [6] Kim,et al, "Block portioning structure in the HEVC standard," IEEE Trans. On circuits and system for video techkology, vol. 22, pp. 1697-1706, Dec. 2012.
- [7] Chih-Ming Fu, Elena Alshina, A. Alshin, Y.W. Huang, C.Y. Chen," Sample Adaptive Offset in the HEVC Standard," IEEE Trans. Circuits and Systems for Video Technology, Vol. 22, No. 12, pp. 1755, Dec. 2012.
- [8] F. Bossen, Et. Al, HEVC complexity and implementation analysis," IEEE Trans-actions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1685-1696, Dec 2012.
- [9] Gary J.Sullivian and Jens-Rainer Ohm. Recent developments in standardization of High Efficiency Video Coding (HEVC) volume 7798.SPIE, 2010.
- [10] T. Wiegand and G. J. Sullivan, "The H.264 video coding standard", IEEE Signal Processing Magazine, vol. 24, pp. 148-153, March 2007.
- [11] G. Sullivan, P. Topiwala and A. Luthra, "The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions", SPIE conference on Applications of Digital Image Processing XXVII, vol. 5558, pp. 53-74, Aug. 2004.