Optical Engineering

SPIEDigitalLibrary.org/oe

Data reuse-based fast subpixel motion estimation for high efficiency video coding

Kiho Choi
Euee S. Jang

Data reuse-based fast subpixel motion estimation for high efficiency video coding

Kiho Choi and Euee S. Jang*
Hanyang University, Digital Media Laboratory, Room 512 R&D Building Hanyang University, 17 Haengdang-Dong, Seongdong-Gu, Seoul 133-791, Republic of Korea

Abstract. A data reuse-based fast subpixel motion estimation (SME) method for high efficiency video coding (HEVC) is proposed. Since SME is one of the most computation-intensive tools in the encoder process, conventional research on SME focused on the reduction of the computational complexity. The applied data-reuse architecture for the design of fast SME substantially reduces computational complexity at the cost of a reasonable increase in memory bandwidth. The core of the proposed data-reuse method is the replacement of redundant computations in SME with the memory access operations of previously computed values. The proposed method was tested in the latest video coding standard, HEVC, with experimental results showing a reduction in operational complexity of ~64.14%, and a reduction in encoding time of ~56.13%, compared to the SME in the HEVC reference encoder. © *2014 Society of Photo-Optical Instrumentation Engineers (SPIE)* [DOI: 10.1117/1.OE.53.6.063103]

Keywords: video coding; high efficiency video coding; fast subpel interpolation; data reuse.

Paper 131836 received Dec. 5, 2013; revised manuscript received May 9, 2014; accepted for publication May 13, 2014; published online Jun. 13, 2014.

1 Introduction

Exponential growth in media content is a driving force in the information technology of today: from broadcasting to point-to-point communication, from real-time to downloading, and from professional production to extemporaneous recording. This rapid growth in media content also implies an increased media usage in diverse environments, such as broadcasting, cable/satellite TV, Internet protocol television, Internet media, and mobile media. Over one billion hits for one song, "Gangnam Style," online demonstrate the manner in which media content can reach a large number of people in a short period of time through diverse media distribution.[1]

Information devices are increasingly equipped with audiovisual services, from traditional TVs and computers to smart phones and tablets. The quality of service requirements for media devices including video resolution, bit rates, frame rates, and codecs is quickly converging toward those previously required only for the "high-end" devices.[2] As a result, the increased complexity of audiovisual codecs imposes a great challenge for media devices. The introduction of the upcoming high efficiency video coding (HEVC) will add substantially to this complexity due to the high compression efficiency of HEVC which is only possible at the cost of increased coding complexity.[3] Increased video resolution (e.g., ultrahigh-definition resolution such as 3840×2160 resolution) in particular is a serious threat to real-time encoding and decoding services.[4]

Many efforts have been made to reduce the computational complexity of the coding process. These methods can be categorized as computation-oriented complexity reduction and memory-oriented complexity reduction. The computation-oriented complexity reduction approaches essentially attempt to minimize the number of instructions within a processor to accomplish a task, whereas the memory-oriented complexity reduction approaches attempt to replace computing instructions with memory-access operations. One example of the computation-oriented complexity reduction approach is the butterfly-based discrete cosine transform (DCT) implementation, which reduces the required computing instructions from 56 additions and 64 multiplications to 29 additions and 12 multiplications in the case of an 8-point one-dimensional DCT.[5] The use of look-up tables in variable-length decoding is a typical example of the memory-oriented complexity reduction approach.[6]

The choice between computation-oriented and memory-oriented approaches is often implementation-dependent, because the given computation architecture will determine which approach is appropriate. The algorithm design should therefore consider implementation aspects to support functionality while maintaining reasonable complexity.[7] Parallel computing based on multicore devices and graphics processing units creates opportunities in the software-based implementation of real-time encoders and decoders. Recently, we summarized the computing and complexity issue in HEVC standard design.[8]

In this article, the data reuse concept in designing fast encoding algorithms is revisited, particularly in fast subpel motion estimation (SME). Past SME algorithms seek primarily to reduce the number of factional-pel search points by efficiently selecting a reduced search area.[9–14] A method using a diamond-shaped search is a typical example of such an algorithm; related algorithms are still being investigated extensively.[13,14] Nevertheless, an investigation into the complexity variation per search point with additional memory usage has not been fully examined so far. Considering the increase of the equipped memory on the computing devices, a fast SME algorithm using additional memory deserves consideration.

The organization of this article is as follows. The motion estimation analysis in HEVC is briefly presented in Sec. 2

*Address all correspondence to: Euee S. Jang, E-mail: esjang@hanyang.ac.kr

0091-3286/2014/$25.00 © 2014 SPIE

and the data reuse-based fast SME method is proposed in Sec. 3. In Sec. 4, the performance of the proposed method is evaluated by the number of operations and running time with an HEVC reference software (HM) encoder. Finally, this article is concluded in Sec. 5.

2 Motion Estimation Analysis in High Efficiency Video Coding

2.1 *Motion Estimation in High Efficiency Video Coding*

HEVC achieves twice the coding efficiency of the previous video coding standard, AVC/H.264, by introducing several new coding tools with a generic coding structure, including the coding unit, prediction unit (PU), and transform unit. PU is a basic unit used for carrying the information related to the prediction processes, and the PU supports various block sizes from 4×4 to 64×64 to find the best block match for the inter prediction.

At each partition in the PU, a two-stage motion estimation process, including an integer-pel search and subpel search, is performed, as shown in Fig. 1. At the beginning of the motion estimation (ME) process, the starting point at each reference frame is set to the position indicated by the best motion-vector predictor using the neighboring PU information. After setting the motion-vector predictor that minimizes the cost of motion-vector coding, the integer-pel search, using search patterns such as the diamond search, is performed. For the half-pel motion estimation, only eight sample points around the position pointed to by the integer-pel accuracy motion vector are searched. The quarter-pel motion vector follows the same procedure used in the half-pel motion-vector search.

An interesting fact with regard to the ME procedure in HEVC is that the number of blocks used for ME is significantly higher than in conventional coding standards. For example, an 8×8 block size is used in the MPEG-2 and MPEG-4 part 2 simple profile and 7 block sizes, 16×16, 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4, are used in the MPEG-4 part10 AVC/H.264. In HEVC, block sizes from 64×64 to 4×4 are allowed when the corresponding coding unit (CU) depth is increased.

When compared to the number of ME partitions in a macroblock of MPEG-4 AVC/H.264, which are the block numbers to be searched at the 16×16 block size, the same size of HEVC is approximately doubled. Due to the increased number of partitions in the ME, the computational complexity increases significantly if every partition mode is tested in the corresponding CU depths. The increased partition numbers result in increased numbers for performing the SME search.

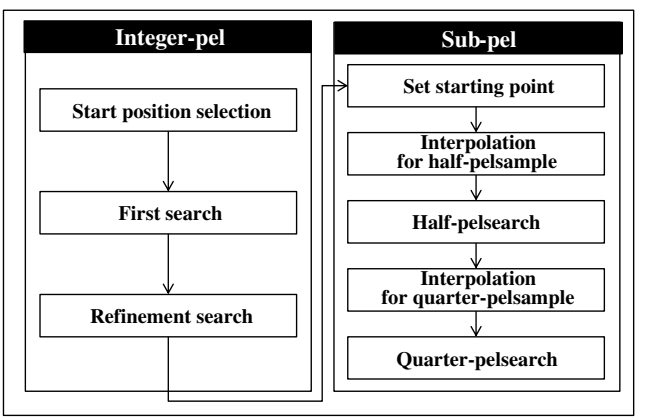


Fig. 1 Motion estimation procedure in high efficiency video coding (HEVC).

2.2 *Interpolation in High Efficiency Video Coding*

In MPEG-4 AVC/H.264, the prediction values at half-pel positions are obtained using a 6-tap Wiener filter, and those at the quarter-pel positions are obtained via a bilinear combination of the samples at the integer and half-pel positions. The interpolation in HEVC directly produces the interpolated values at a fractional accuracy from the samples at integer-pel positions by deriving the values via a DCT-based interpolation filter (DCT-IF), using the filter coefficients of an 8-tap s shown in Table 1.

DCT-IF uses a higher number of operations when compared to that of MPEG-4 AVC/H.264, because of the increased tap size required for the change from the 6-tap filter to the 8-tap filter for half-pel samples and the binary filter to the 7-tap filter for quarter-pel samples. Figure 2 and Table 2 show the details of DCT-IF. In Fig. 2, the shaded blocks contain integer samples. Upper-case letters and fractional sample positions are shown in the un-shaded blocks,

Table 1 Coefficients for 8-tap DCT-based interpolation filter.

Position	Filter coefficients
1/4	{−1, 4, −10, 58, 17, −5, 1}
2/4	{−1, 4, −11, 40, 40, −11, 4, −1}
3/4	{1, −5, 17, 58, −10, 4, −1}

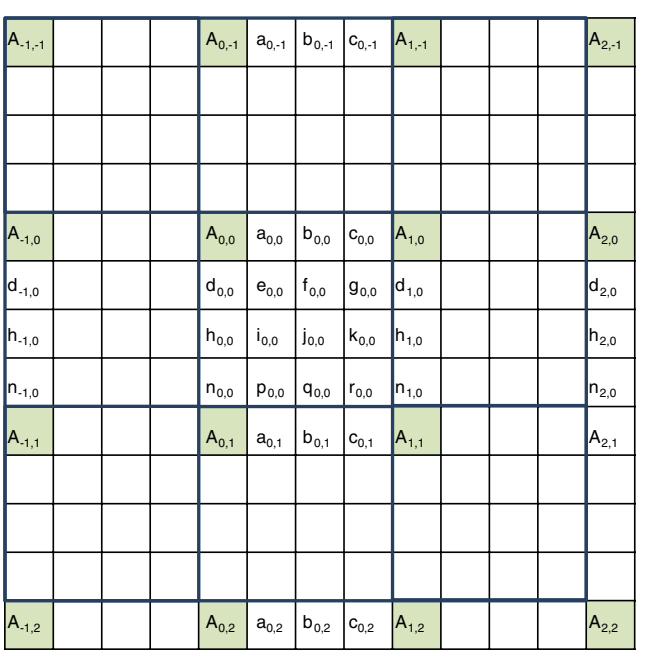


Fig. 2 Integer samples (shaded blocks with upper-case letters) and fractional sample positions (unshaded blocks with lower-case letters) for quarter-sample luma interpolation.

Table 2 Equation to generate interpolated samples given the luma integer samples: (a) The positions for interpolation and input values, (b) the block diagram of each interpolation.

		Input							
Position	Type	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8
$a_{0,0}$	Quarter-pel_A	$A_{-3,0}$	$A_{-2,0}$	$A_{-1,0}$	$A_{0,0}$	$A_{1,0}$	$A_{2,0}$	$A_{3,0}$	N/A
$b_{0,0}$	Half-pel	$A_{-3,0}$	$A_{-2,0}$	$A_{-1,0}$	$A_{0,0}$	$A_{1,0}$	$A_{2,0}$	$A_{3,0}$	$A_{4,0}$
$c_{0,0}$	Quarter-pel_B	$A_{-2,0}$	$A_{-1,0}$	$A_{0,0}$	$A_{1,0}$	$A_{2,0}$	$A_{3,0}$	$A_{4,0}$	N/A
$d_{0,0}$	Quarter-pel_A	$A_{0,-3}$	$A_{0,-2}$	$A_{0,-1}$	$A_{0,0}$	$A_{0,1}$	$A_{0,2}$	$A_{0,3}$	N/A
$h_{0,0}$	Half-pel	$A_{0,-3}$	$A_{0,-2}$	$A_{0,-1}$	$A_{0,0}$	$A_{0,1}$	$A_{0,2}$	$A_{0,3}$	$A_{0,4}$
$n_{0,0}$	Quarter-pel_B	$A_{0,-2}$	$A_{0,-1}$	$A_{0,0}$	$A_{0,1}$	$A_{0,2}$	$A_{0,3}$	$A_{0,4}$	N/A
$e_{0,0}$	Quarter-pel_A	$a_{0,-3}$	$a_{0,-2}$	$a_{0,-1}$	$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	N/A
$i_{0,0}$	Half-pel	$a_{0,-3}$	$a_{0,-2}$	$a_{0,-1}$	$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$
$p_{0,0}$	Quarter-pel_B	$a_{0,-2}$	$a_{0,-1}$	$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	N/A
$f_{0,0}$	Quarter-pel_A	$b_{0,-3}$	$b_{0,-2}$	$b_{0,-1}$	$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	$b_{0,3}$	N/A
$j_{0,0}$	Half-pel	$b_{0,-3}$	$b_{0,-2}$	$b_{0,-1}$	$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	$b_{0,3}$	$b_{0,4}$
$q_{0,0}$	Quarter-pel_B	$b_{0,-2}$	$b_{0,-1}$	$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	$b_{0,3}$	$b_{0,4}$	N/A
$g_{0,0}$	Quarter-pel_A	$c_{0,-3}$	$c_{0,-2}$	$c_{0,-1}$	$c_{0,0}$	$c_{0,1}$	$c_{0,2}$	$c_{0,3}$	N/A
$k_{0,0}$	Half-pel	$c_{0,-3}$	$c_{0,-2}$	$c_{0,-1}$	$c_{0,0}$	$c_{0,1}$	$c_{0,2}$	$c_{0,3}$	$c_{0,4}$
$r_{0,0}$	Quarter-pel_B	$c_{0,-2}$	$c_{0,-1}$	$c_{0,0}$	$c_{0,1}$	$c_{0,2}$	$c_{0,3}$	$c_{0,4}$	N/A

and lower-case letters represent a quarter-sample luma interpolation. The fractional sample values in each position are derived via the equation in Table 2.

2.3 *Computational Analysis of Interpolation*

The use of variable block sizes and subpel ME brings a substantial improvement in coding efficiency, but adds significant computational complexity in the encoding process. The subpixel interpolation in ME in particular presents a challenge in the encoder implementation process. Figure 3 shows a profiling result of HM encoding time with the random-access configuration. In the figure, a significant portion of the encoding time is spent in the interpolation process. Considering that the interpolation process is used only for SME, the amount of time for interpolation filtering is significantly high, at ~20% of the encoding time and approximately two-thirds of the SME process.

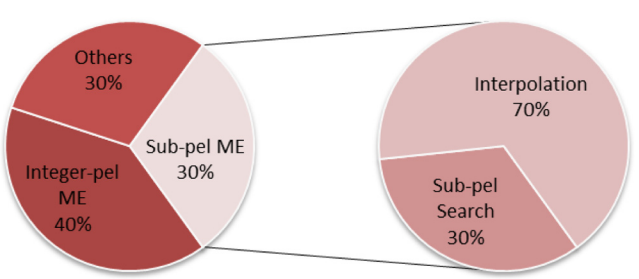


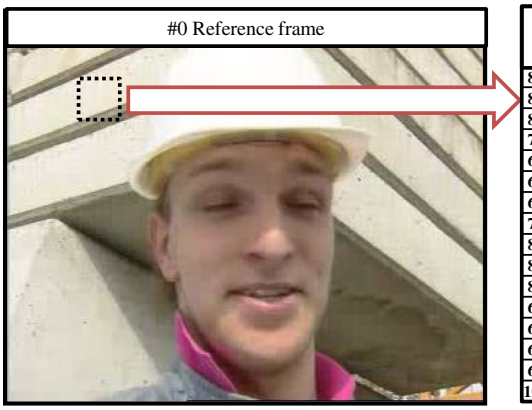
Fig. 3 Time spent for motion estimation in the HEVC reference software encoder.

Reducing the operational complexity of interpolation filtering is difficult, as it is close to optimal within the algorithm definition. Thus, the conventional approach to reducing computational complexity is increasing processing capability, using single instruction, multiple data (SIMD) optimization, hardware accelerators, reducing executed instruction, and parallel computing. These methods are limited by the usage of an external capability, which does not lessen the computational complexity of interpolation. Thus, it is required that each interpolation addresses a reasonable complexity of implementation aspects.

3 Proposed Method

3.1 *Motivation*

As analyzed in the previous section, one of the most significant computations in SME occurs during the interpolation process to compute the subpel values. Note that the interpolation filtering is performed whenever a context is changed, such as a PU partition or a reference-frame number update. This requirement is compounded because the same sample generation repeats at the same position each time, increasing complexity. As shown in Fig. 4, a typically referenced block may be regenerated or interpolated up to 107 times, while producing exactly the same reference-interpolated pixel values. For the test, we used the test zone search ME algorithm in a low-delay configuration for HEVC development.[15] This problem demonstrates how an algorithm designed without



The number of selections for best integer-pel MV at x=76-92 and y=16-32

85	86	82	78	80	77	81	93	99	97	95	93	86	87	90	93	96	94	92
85	86	82	78	80	77	81	93	99	97	95	93	86	87	90	93	96	94	92
85	86	82	78	80	77	81	93	99	97	95	93	86	87	90	93	96	94	92
73	72	70	62	64	63	64	78	90	88	85	78	68	67	74	78	80	80	76
68	66	64	58	64	62	63	75	87	88	85	80	70	67	74	78	80	80	76
68	66	64	58	64	62	63	75	87	88	85	80	70	67	74	78	80	80	76
68	66	64	58	64	62	63	75	87	88	85	80	70	67	74	78	80	80	76
77	77	74	69	63	63	62	75	85	84	84	84	78	75	82	78	88	88	84
80	78	74	74	70	73	73	87	97	92	91	86	78	75	82	78	88	88	84
80	78	74	74	70	73	73	87	97	92	91	86	78	75	82	78	88	88	84
80	78	74	74	70	73	73	87	97	92	91	86	78	75	82	78	88	88	84
66	64	60	62	60	61	65	78	87	84	82	73	65	62	66	64	74	74	70
64	62	58	56	54	55	59	72	81	77	75	70	62	62	66	64	74	74	70
64	62	58	56	54	55	59	72	81	77	75	70	62	62	66	64	74	74	70
66	64	59	57	55	56	60	73	82	78	75	70	62	62	66	64	74	74	70
102	100	99	94	87	88	112	107	105	105	95	92	81	79	91	92	91	89	86

Fig. 4 The number of selections for the best integer-pel motion vector, where $x = 76$ to 92 and $y = 16$ to 32 in reference frame #0.

consideration of implementation aspect can limit complexity reduction, since little complexity reduction is possible if the reduction approach is limited to an algorithm-oriented tool.

As an alternative to the algorithm-oriented tool, we apply an analysis of the practical SME operation for complexity reduction in SME using a data-reuse method. Fundamentally, the idea is to replace redundant computations with memory-access operations, as shown in Table 3. As seen in this table, the benefit of the data-reuse–based algorithm is clear: it reduces the number of operations from 138 to 11. Use of this algorithm comes at the cost of increased memory size; overall performance is, therefore, dependent on how well one can harness the algorithm by balancing the computation and memory operations.

3.2 *Core Algorithm*

An overview of the proposed method is depicted in Fig. 5. The output of the proposed SME consists of interpolated samples with the best motion vector (MV) where the current PU partition is applied. The interpolated samples from our proposed method match those of the current SME in HM 8.0.

The process begins to set a starting point using integer-pel MV information. If the position from the integer-pel MV search has been previously referred, the proposed method skips the computations for the half-pel interpolation samples. In addition, the proposed method includes the process of copying the previous half-pel interpolation samples from the data stored in the table. Otherwise, the process for half-pel interpolation is same as the original half-pel interpolation procedure in HEVC, generating half-pel interpolation samples used by an 8-tap DCT-IF. The only difference is that

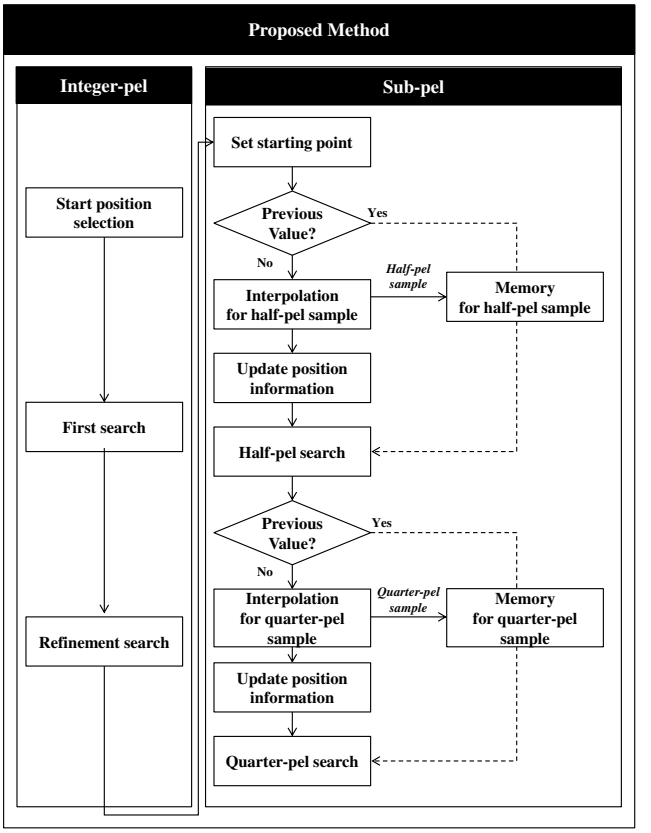


Fig. 5 Block diagram of the proposed subpel ME based on data-reuse architecture.

Table 3 The number of operations for a half-pel and quarter-pel interpolations corresponding to an integer pixel (MUL: # multiplication, ADD: # additions, SHIFT: # shift operations, MA: # memory accesses).

	Operation				
Type	MUL	ADD	SHIFT	MA	SUM
High efficiency video coding interpolation	58	69	11	0	138
Data reuse-based interpolation	0	0	0	11	11

the encoder updates the current integer-pel information for the next subpel ME by describing the current position and the interpolated samples to the information map and the interpolated sample table. The proposed method for the quarter-pel interpolation follows the same procedure proposed in the half-pel interpolation. When compared to the conventional interpolation process, the proposed method reduces computational redundancy by adaptively including the interpolation calculation and employing a table look-up to eliminate operations in interpolation.

To check whether the position was referred during the previous SME, the proposed method uses an information

map for half-pel and quarter-pel interpolations. The maps corresponding to integer-pel MV and half-pel MV information consider three factors: the MV position, the size of the PU partition mode, and the reference frames. After calculating the interpolation samples, the proposed method makes an entry of the MV position and the size of the PU partition mode at the map of each reference frame. The proposed method preserves the interpolated sample table, which includes all of the interpolated samples with regard to each reference frame after calculating the interpolation samples provided in a normal interpolation process. The information map and the interpolated sample table are refreshed when the reference frames change.

The proposed data-reuse method efficiently reduces computations by replacing them with the process of checking previous information, whether or not the position was previously referred. The checking operation significantly reduces the number of operations that are regarding with multiple PU modes locate the same best-MV position. The proposed method works efficiently for the HEVC architecture due to the fact that the standard has a higher chance of finding the same position as the best-MV position with the increased number of ME partitions and accurate motion-vector coding.

3.3 *Computational Complexity of the Proposed Method*

The computational complexity based on the proposed method can be described as follows:

$$C_{\text{pro}} = \alpha * C_{\text{ma}} + (1 - \alpha) * C_{\text{oper}} + C_{\text{over}}, \tag{1}$$

where C_{pro} is the average computational complexity for a pixel interpolation, C_{oper} is the operational complexity for a pixel interpolation, C_{ma} is the memory access complexity based on the use of the proposed data-reuse interpolation for a pixel, C_{over} is the average overhead complexity based on the use of the proposed data-reuse interpolation for a pixel, and α is the average probability applied to the proposed data-reuse interpolation.

Higher α results in less C_{pro}, as C_{ma} has less computational complexity than C_{oper}, as shown in Table 3. This choice should be carefully considered, since computation reduction is obtained at the cost of additional memory usage. If an encoding environment contains sufficient memory, the tradeoff is not a problem. When memory resources are limited, it is necessary to locate the optimal point, less than α, at the cost of a reasonable increase in memory bandwidth.

The memory usage of the proposed method can be described as follows:

$$M_{\text{pro}} = \alpha * M_{\text{add}} + (1 - \alpha) * M_{\text{org}}, \tag{2}$$

where M_{pro} is the average memory usage for the proposed method, M_{add} is the average additional memory usage from the data-reuse–based interpolation, M_{org} is the average memory usage for the original interpolation, and α is the average probability applied to the proposed data-reuse interpolation.

In an environment with limited memory resources, it is possible to select the optimal value, α', between computational complexity and memory usage as follows:

$$\alpha' = \arg \min_{\alpha' \le \alpha} \{C_{\text{pro}} + \lambda M_{\text{pro}}\}, \tag{3}$$

where λ is a Lagrange multiplier for the optimal selection. The value, λ, can vary based on the encoding environment. Equation (3) provides valuable implementation aspect data, aiding the algorithm designer in the reduction of computational complexity within a limited memory resource.

4 Experimental Results

To determine the performance of the proposed method, we evaluated the reduction in operations of the proposed SME based on data-reuse architecture and compared it to that of HM 8.0. Additionally, the encoding time to confirm the comparison in operational complexity using HM 8.0 is evaluated. The details of the encoding configurations are described in Table 4.

4.1 *Computational Complexity Evaluation of the Proposed Method*

In order to evaluate the reduction in computational complexity based on the proposed method, we measured the number of additions, multiplications, and memory accesses. The proposed method reduces the number of operations by replacing the calculations of the interpolation with memory accesses for reading the previously interpolated samples in tables; thus, the number of memory accesses is included for the table look-up when counting the number of operations. In the experiment, we assume that the computational complexity of memory access is approximately the same as that of the other operations and the overhead of the proposed method, C_{over}, is trivial and not necessary to optimal selection between computation reduction and memory usage because of sufficient memory resources (i.e., 8 GB of RAM).

Based on the number of pixels applied to the proposed method, the reduction in computations for SME was evaluated and the reduction ratio was based on the following equation:

Table 4 Test conditions and software reference configurations.

Test sequences	■ Class B (1920 × 1080): *ParkScene*, *Cactus*, and *QTerrace*
	■ Class C (832 × 480): *BasketballDrill*, *BQMall*, and *RaceHorsesC*
	■ Class D (416 × 240): *BasketballPass*, *BQSquare*, and *RaceHorses*
	■ Class E (1280 × 720): *FourPeople, Johnny, and KristenAndSara*
Total frames	■ 10 seconds of video duration
Software	■ HM 8.0[16]
Quantization parameter	■ 22, 27, 32, and 37
Others	■ Low delay, high-efficiency setting[15]

$$\Delta\,\text{Ratio} = \frac{C_{\text{PRO}} - C_{\text{REF}}}{C_{\text{REF}}} \times 100. \quad (4)$$

Tables 5 and 6 show the results of the computational complexity. As shown, the computational complexity of the proposed method is significantly lower than that for interpolation in HM 8.0. Specifically, the number of operations in the proposed method is ~62.27% lower than that of HM 8.0 in Classes B and C, and ~66.01% lower than that of HM 8.0 in Classes D and E, respectively. The best performance is shown in Class E sequences: a reduction of ~75.1% compared to that of HM 8.0. For the worst case, Class D sequences require ~56.92% fewer operations compared to that of HM 8.0. The results show that the data reuse architecture is more effective when the test sequence contains less motion activity. Through the experiments, the proposed data-reuse architecture

Table 5 Comparison of the number of operations for Class B and Class C (MUL: the billion number of multiplication, ADD: the billion number of additions, SHIFT: the billion number of shift operations, MA: the billion number of memory accesses, SUM: the sum of operations).

			HM 8.0				Proposed method					
	Sequence	QP	MUL	ADD	SHIFT	SUM	MUL	ADD	SHIFT	MA	SUM	Δ Ratio (%)
Class B	ParkScene	22	2974	3538	564	**7076**	916	1091	175	389	**2572**	−63.65
		27	2790	3319	529	**6638**	901	1073	172	357	**2503**	−62.30
		32	2658	3162	504	**6324**	901	1073	172	332	**2478**	−60.82
		37	2569	3057	487	**6113**	912	1086	174	313	**2486**	−59.33
	Cactus	22	6016	7157	1141	**14,314**	1654	1970	316	825	**4764**	−66.72
		27	5664	6738	1074	**13,476**	1445	1722	276	798	**4241**	−68.53
		32	5513	6559	1046	**13,117**	1398	1665	267	779	**4108**	−68.68
		37	5388	6410	1022	**12,819**	1383	1646	264	758	**4051**	−68.40
	BQTerrace	22	7664	9117	1453	**18,234**	2493	2969	476	977	**6916**	−62.07
		27	6903	8212	1309	**16,424**	1940	2310	370	939	**5559**	−66.15
		32	6516	7752	1236	**15,504**	1779	2119	339	897	**5134**	−66.89
		37	6346	7550	1204	**15,100**	1740	2071	331	872	**5014**	−66.79
						Class B average						−65.03
Class C	BasketballDrill	22	1181	1405	224	**2810**	326	388	62	162	**937**	−66.65
		27	1142	1359	217	**2717**	338	402	64	152	**956**	−64.81
		32	1108	1318	210	**2636**	343	408	65	145	**961**	−63.56
		37	1077	1282	204	**2563**	330	393	63	141	**928**	−63.79
	BQMall	22	1445	1719	274	**3438**	414	493	79	195	**1181**	−65.67
		27	1382	1644	262	**3288**	402	479	77	185	**1143**	−65.24
		32	1331	1583	252	**3167**	394	470	75	177	**1117**	−64.74
		37	1288	1533	244	**3065**	392	467	75	169	**1103**	−64.00
	RaceHorses	22	773	919	147	**1839**	355	423	68	79	**924**	−49.75
		27	748	890	142	**1781**	347	414	66	76	**903**	−49.30
		32	721	858	137	**1716**	339	404	65	72	**880**	−48.69
		37	692	823	131	**1646**	332	395	63	68	**857**	−47.91
						Class C average						−59.51
						Total average						−62.27

Table 6 Comparison of the number of operations for Class D and Class E (MUL: the billion number of multiplication, ADD: the billion number of additions, SHIFT: the billion number of shift operations, MA: the billion number of memory accesses, SUM: the sum of operations).

			HM 8.0				Proposed method					
Class	Sequence	QP	MUL	ADD	SHIFT	SUM	MUL	ADD	SHIFT	MA	SUM	Ratio (%)
Class D	BasketballPass	22	289	344	55	**687**	106	126	20	35	**286**	–58.35
		27	282	336	54	**671**	107	127	20	33	**287**	–57.23
		32	274	326	52	**653**	107	127	20	32	**286**	–56.18
		37	264	314	50	**628**	105	125	20	30	**280**	–55.44
	BQSquare	22	370	440	70	**881**	106	127	20	50	**303**	–65.60
		27	359	427	68	**854**	106	126	20	48	**300**	–64.87
		32	339	403	64	**806**	102	122	19	45	**288**	–64.26
		37	312	371	59	**741**	95	113	18	41	**268**	–63.92
	RaceHorses	22	186	221	35	**443**	84	99	16	19	**218**	–50.71
		27	182	216	35	**433**	83	99	16	19	**217**	–49.94
		32	176	209	33	**418**	82	98	16	18	**214**	–48.81
		37	167	199	32	**397**	80	96	15	16	**208**	–47.76
						Class D average						–56.92
Class E	FourPeople	22	2982	3548	566	**7096**	481	573	92	474	**1619**	–77.18
		27	2904	3455	551	**6910**	417	496	80	471	**1464**	–78.82
		32	2860	3403	542	**6806**	389	463	74	468	**1394**	–79.52
		37	2829	3365	536	**6731**	376	448	72	465	**1360**	–79.79
	Johnny	22	2968	3531	563	**7062**	652	776	124	439	**1991**	–71.81
		27	2868	3412	544	**6824**	493	586	94	450	**1623**	–76.22
		32	2826	3361	536	**6723**	440	524	84	452	**1500**	–77.69
		37	2803	3335	532	**6670**	412	491	78	453	**1435**	–78.49
	KristenAndSara	22	2983	3549	566	**7098**	795	946	152	414	**2306**	–67.51
		27	2896	3445	549	**6890**	688	819	131	418	**2056**	–70.16
		32	2845	3385	540	**6770**	637	758	121	418	**1934**	–71.43
		37	2817	3352	534	**6703**	597	710	113	421	**1841**	–72.54
						Class E average						–75.10
						Total average						–66.01

significantly reduces the number of operations for all test sequences, regardless of quantization parameter (QP) values.

4.2 *Evaluation of the Encoding Time and Motion Estimation Time*

In order to confirm the reduction in operations, the total encoding time and ME time in the encoder are measured when compared to HM 8.0. The test was also conducted on a PC with 8 GB of RAM. The reduction ratio was evaluated based on the following equation:

$$\Delta\,\text{Time} = \frac{T_{\text{PRO}} - T_{\text{REF}}}{T_{\text{REF}}} \times 100. \tag{5}$$

Tables 7 and 8 show the total encoding time and ME time of the proposed method, as well as those of HM 8.0. As

Table 7 Comparison of encoding time and ME time for Class B and Class C.

			HM 8.0		Proposed method		Comparison	
	Sequence	QP	Total time (s)	ME time (s)	Total time (s)	ME time (s)	Δ Total time (%)	Δ ME time (%)
Class B	ParkScene	22	28,875	11,227	24,501	6693	−15.15	−40.38
		27	24,111	10,549	20,014	6380	−16.99	−39.52
		32	21,102	10,040	17,364	6182	−17.71	−38.43
		37	19,262	9696	15,741	6061	−18.28	−37.49
	Cactus	22	60,513	22,512	51,364	13,127	−15.12	−41.69
		27	47,973	21,096	39,136	12,031	−18.42	−42.97
		32	42,595	20,516	33,879	11,666	−20.46	−43.14
		37	39,156	20,019	30,697	11,404	−21.60	−43.03
	BQTerrace	22	85,348	28,980	73,988	17,596	−13.31	−39.28
		27	56,467	25,945	45,746	14,965	−18.99	−42.32
		32	47,965	24,474	37,795	14,041	−21.20	−42.63
		37	44,517	23,858	34,518	13,666	−22.46	−42.72
			Class B average				−18.31	−41.13
Class C	BasketballDrill	22	9533	4493	8626	1908	−9.52	−57.53
		27	8126	4371	7273	1858	−10.50	−57.49
		32	6999	4219	6201	1829	−11.40	−56.65
		37	6212	4085	5430	1762	−12.58	−56.86
	BQMall	22	10,707	5510	9607	2305	−10.27	−58.17
		27	9003	5288	7964	2218	−11.54	−58.05
		32	7901	5087	6907	2137	−12.58	−57.99
		37	7153	4945	6193	2082	−13.42	−57.90
	RaceHorses	22	7997	2988	7574	1457	−5.28	−51.24
		27	6664	2850	6257	1414	−6.12	−50.39
		32	5636	2745	5257	1345	−6.72	−51.00
		37	4850	2627	4497	1286	−7.27	−51.05
			Class C average				−9.77	−55.36
			Total average				−14.04	−48.25

expected, the proposed method significantly outperformed the HM 8.0. Specifically, when compared to HM 8.0, the total encoding time of the proposed method is, on average, 14.04% lower, and the ME time of the proposed method is, on average, 48.25% lower. In the optimal case, the total encoding time of the proposed method was 18.56% lower than that of HM 8.0 at Class E sequences, and the best ME performance time shows that the proposed method is 66.58% lower when compared to HM 8.0 at Class D sequences.

Considering that the time needed for SME is approximately one-third of the ME process, the experimental results shown above clearly demonstrate that the proposed data-reuse method can efficiently reduce computational redundancy in the SME process.

For the performance of the proposed method, it is important to determine how to choose the tradeoff between a memory access and a computation. In the implementation, we increase the memory size four times for the process of ME. Although the increased memory usage may look heavier, the speedup factor is more important for some applications such as real-time encoding and broadcasting. The speedup from the proposed method is close to twice as

Table 8 Comparison of encoding time and ME time for Class D and Class E.

			HM 8.0		Proposed method		Comparison	
Class	Sequence	QP	Total time (s)	ME time (s)	Total time (s)	ME time (s)	Δ Total time (%)	Δ ME time (%)
Class D	BasketballPass	22	2560	1091	2372	507	−7.37	−53.52
		27	2233	1066	2056	497	−7.93	−53.37
		32	1950	1037	1787	484	−8.38	−53.34
		37	1713	998	1553	465	−9.35	−53.41
	BQSquare	22	2925	1408	2632	598	−10.02	−57.53
		27	2294	1382	2016	576	−12.13	−58.32
		32	1889	1320	1634	548	−13.52	−58.48
		37	1630	1216	1399	504	−14.15	−58.55
	RaceHorses	22	1858	2988	1761	350	−5.24	−88.29
		27	1595	2850	1502	343	−5.82	−87.96
		32	1352	2745	1266	328	−6.32	−88.05
		37	1155	2627	1074	313	−6.98	−88.09
			Class D average				−8.93	−66.58
Class E	FourPeople	22	15,804	11,238	13,038	4218	−17.50	−62.47
		27	14,017	10,882	11,294	4067	−19.42	−62.63
		32	13,158	10,713	10,487	3925	−20.30	−63.36
		37	12,611	10,654	9921	3879	−21.33	−63.59
	Johnny	22	15,198	11,255	12,677	4430	−16.59	−60.64
		27	13,382	10,777	10,782	4108	−19.43	−61.88
		32	12,638	10,619	10,035	3967	−20.59	−62.64
		37	12,274	10,383	9611	3946	−21.70	−62.00
	KristenAndSara	22	16,694	11,116	14,271	4686	−14.51	−57.84
		27	14,655	10,764	12,299	4374	−16.08	−59.36
		32	13,603	10,655	11,277	4258	−17.10	−60.04
		37	12,964	10,050	10,615	4183	−18.11	−58.38
			Class E average				−18.56	−61.24
			Total average				−13.55	−64.01

fast as the current method in the SME of HEVC. Therefore, we believe that it is worth the cost of a reasonable increase in memory.

5 Conclusion

In this article, a data-reuse–based is introduced, the fast SME algorithm that considers the choice between the computation-oriented and memory-oriented approaches in order to maximize functionality while maintaining reasonable complexity. By applying this concept to the recent coding standard, SME of HEVC, the proposed method substantially reduces computational complexity in the ME process. The experimental results show that the proposed method reduces operational complexity by ∼64.14% and reduces ME time by ∼56.13% when compared to HM 8.0.

Acknowledgments

This work was supported by the Industrial Strategic Technology Development Program (10047438, Development and international standardization of MPEG Type-1 standard

technology) funded by the Ministry of Trade, Industry and Energy (MOTIE) of Korea.

References

1. D. Bevan, "A year after 'Gangnam Style,' K-pop continues to make its mark in America," Washington Post, http://www.washingtonpost.com/entertainment/music/a-year-after-gangnam-style-k-pop-continues-to-make-its-mark-in-america/2013/11/07/cb161c56-431f-11e3-a751-f032898f2dbc_story.html (9 November 2013).
2. W. Hwang and P. Tseng, "A QoS-aware residential gateway with bandwidth management," *IEEE Trans. Consum. Electron.* **51**(3), 840–848 (2005).
3. T. Wiegand et al., "Special section on the joint call for proposals on high efficiency video coding (HEVC) standardization," *IEEE Trans. Circuits Syst. Video Technol.* **20**(12), 1661–1666 (2010).
4. X. Peng et al., "Highly parallel line-based image coding for many cores," *IEEE Trans. Image Proc.* **21**(1), 196–206 (2012).
5. H. S. Hou, "A fast recursive algorithm for computing the discrete cosine transform," *IEEE Trans. Acoust. Speech, Signal Process.* **35**(10), 1455–1461 (1987).
6. T. C. Chen et al., "Architecture design of context-based adaptive variable-length coding for H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.* **53**(9), 832–836 (2006).
7. G. G. Lee et al., "Algorithm/architecture co-exploration of visual computing on emergent platforms: overview and future prospects," *IEEE Trans. Circuits Syst. Video Technol.* **19**(11), 455–465 (2009).
8. K. Choi and E. S. Jang, "Leveraging parallel computing in modern video coding standards," *IEEE MultiMedia* **19**(3), 7–11 (2012).
9. Z. Chen, P. Zhou, and Y. He, "Fast integer-pixel and fractional pel motion estimation for JVT," document JVT-F017, ITU-T, Awaji Island, Japan (2002).
10. X. Xu and Y. He, "Improvements on fast motion estimation strategy for H. 264/AVC," *IEEE Trans. Circuits Syst. Video Technol.* **18**(3), 285–293 (2008).
11. J. W. Suh and J. Jechang, "Fast sub-pixel motion estimation techniques having lower computational complexity," *IEEE Trans. Consum. Electron.* **50**(3), 968–973 (2004).
12. W. Lin et al., "A new class based early termination method for fast motion estimation in video coding," in *Proc. of the IEEE Int. Symposium on Circuits Systems*, pp. 625–628, IEEE (2009).
13. L. Yang et al., "Prediction-based directional fractional pixel motion estimation for H.264 video coding," in *Proc. IEEE Int. Conf. Acoustics, Speech Signal Processing*, Vol. 2, pp. 901–904, IEEE (2005).
14. Y. Shen et al., "Improved fractional pixel motion estimation algorithm for H. 264/AVC," in *Proc. IEEE Int. Conf. Signal Process.*, Vol. 2, IEEE (2006).
15. F. Bossen, "Common test conditions and software reference configurations," document JCTVC-K1100 of JCT-VC, Shanghai, China (2012).
16. High Efficiency Video Coding Test Model software 8.0, https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware.

Kiho Choi received his BS degree (summa cum laude) in information and communication engineering and MS and PhD degrees from the Department of CSE at Hanyang University in Seoul, Republic of Korea in 2008, 2010, and 2012, respectively. He received the Best Paper Award from Hanyang University in 2012. He is a postdoctoral researcher at the same university and serving as the chair of MPEG ad-hoc group on Internet video coding. His research interests include image and video compression, and MPEG standardization.

Euee S. Jang received his BS degree from Jeonbuk National University, Republic of Korea, and a PhD from SUNY at Buffalo, New York. He is a professor at the Department of CSE, Hanyang University, Seoul, Republic of Korea. His research interests include image/video coding and reconfigurable video coding. He has authored more than 150 MPEG contribution papers, more than 30 journal or conference papers, and two book chapters. He has received three ISO/IEC Certificates of Appreciation for the contribution in MPEG-4 development.