A Fast Sub-Pixel Motion Estimation Algorithm for H.264/AVC Video Coding

Weiyao Lin, Krit Panusopone, David M. Baylon, Ming-Ting Sun, *Fellow, IEEE,* Zhenzhong Chen, *Member, IEEE,* and Hongxiang Li, *Senior Member, IEEE*

Abstract—**Motion estimation (ME) is one of the most time-consuming parts in video coding. The use of multiple partition sizes in H.264/AVC makes it even more complicated when compared to ME in conventional video coding standards. It is important to develop fast and effective sub-pixel ME algorithms since: 1) the computation overhead by sub-pixel ME has become relatively significant while the complexity of integer-pixel search has been greatly reduced by fast algorithms, and 2) reducing sub-pixel search points can greatly save the computation for sub-pixel interpolation. In this letter, a novel fast sub-pixel ME algorithm is proposed which performs a "rough" sub-pixel search before the partition selection, and performs a "precise" sub-pixel search for the best partition. By reducing the searching load for the large number of non-best partitions, the computation complexity for sub-pixel search can be greatly decreased. Experimental results show that our method can reduce the sub-pixel search points by more than 50% compared to existing fast sub-pixel ME methods with negligible quality degradation.**

Index Terms—**Fast algorithm, sub-pixel motion estimation.**

I. Introduction

H.264/AVC is the state-of-the-art video coding standard established by ITU-T and ISO/IEC. H.264/AVC uses many new techniques and is able to save more than 50% in bitrate (BR) while having similar video quality compared to the MPEG-2 video coding standard [1].

Manuscript received June 23, 2010; revised August 21, 2010; accepted September 28, 2010. Date of publication January 17, 2011; date of current version March 2, 2011. This work was supported in part by the Chinese National 973, under Grants 2010CB731401 and 2010CB731406, in part by the National Science Foundation of China, under Grants 60632040, 60928003, 60933006, 60973067, and 61001146, and in part by the National Science Foundation of USA, under Grant 1032567. The main part of this work was performed when W. Lin was employed with Motorola. This paper was recommended by Associate Editor R. Lukac.

W. Lin is with the Institute of Image Communication and Information Processing, Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: wylin@sjtu.edu.cn).

K. Panusopone and D. M. Baylon are with the Department of Advanced Technology, Mobile Devices and Home, Motorola Inc., San Diego, CA 92121 USA (e-mail: krit@motorola.com; david.baylon@motorola.com).

M.-T. Sun is with the Department of Electrical Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: mts@u.washington.edu).

Z. Chen is with the School of Electrical and Electronic Engineering, Nanyang Technological University, 639798, Singapore (e-mail: zzchen@ntu.edu.sg).

H. Li is with the Department of Electrical and Computer Engineering, North Dakota State University, Fargo, ND 58108 USA (e-mail: hongxiang.li@ndsu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2011.2106290

Motion estimation (ME) is one of the most time-consuming parts in video coding. Developing fast algorithms for ME to reduce computational complexity in video coding has been an important and challenging problem. In the H.264/AVC joint model (JM) [5], the ME process contains two stages: integer pixel search over a large area and sub-pixel search around the best selected integer pixel. Since H.264/AVC uses seven partition sizes for inter-frame prediction (16×16, 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4), the complexity of multi-partition ME is high [2]. It is becoming more critical to develop fast and effective sub-pixel ME algorithms for H.264/AVC. First, the computation overhead by sub-pixel ME has become relatively significant while the complexity of integer-pixel search has been greatly reduced by fast algorithms. For example, there have been integer-pixel ME algorithms [4], [10], [16] that only need between three and five integer search points to calculate the final integer motion vector (MV). The computation in the 16-point sub-pixel search method used in the JM thus becomes comparatively large. Second, typical sub-pixel searches require interpolating sub-pixel values for computing the sum of absolute difference (SAD). Reducing sub-pixel search points can also reduce the interpolation computation time.

In this letter, a novel sub-pixel ME algorithm is proposed for H.264/AVC, which performs a "rough" sub-pixel search before the partition selection, and performs a "precise" sub-pixel search for the best partition. By reducing the searching load for the large number of non-best partitions, the computation complexity for sub-pixel search can be greatly decreased. Experimental results show that the proposed algorithm can significantly reduce the number of sub-pixel search points compared to other fast sub-pixel ME algorithms [6]–[9], with negligible quality degradation.

The remainder of this letter is organized as follows. Section II reviews existing research on sub-pixel ME. Section III provides in-depth analysis on how to further reduce the search points for sub-pixel ME for multiple partitions. The proposed algorithm is described in Section IV. Section V shows the experimental results and Section VI concludes this letter.

II. Related Work

Chen *et al.* [6] analyzed the difference between the integer-pixel matching error surface and the sub-pixel matching error surface. According to Chen's analysis, the integer-pixel matching error surface is far from a unimodal surface inside the searching window due to the complexity of the video content. The assumption of unimodal will easily result in trapping in a local minimum. However, for the sub-pixel matching error surface, the unimodal surface assumption holds in most cases because of the smaller search range of sub-pixel ME as well as the high correlation between sub-pixels due to the sub-pixel interpolation.

There has been much research on fast sub-pixel ME [6]–[9], [17]. Most of these methods are based on the unimodal surface assumption and perform the sub-pixel search in two steps as follows:

1) predict a sub-pixel MV (*SPMV*);
2) perform a small area search around the *SPMV* to obtain the final SPMV.

The method to get the sub-pixel predicted MV can be summarized in two ways: using spatiotemporal information and modeling the SAD surface.

Chen *et al.* [6] and Yang *et al.* [8] used spatiotemporal information to get the SPMVs. In [6], a center-biased fractional pixel search (CBFPS) fast sub-pixel ME method is studied, where the MVs of neighboring MBs were used to get the SPMV as follows:

$$\text{SPMV} = (pred\_mv - MV)\%\beta \tag{1}$$

where *pred_mv* is the MV prediction of the current partition (in sub-pixel resolution), *MV* is the best integer-pixel MV of the current partition ($\beta = 4$ in the 1/4-pixel case and $\beta = 8$ in the 1/8-pixel case), and % represents the modulo operation. In [8], a larger partition MV (e.g., 16×8 inter-mode MV takes a 16×16 MV as a reference) or previous frame MV was used to get the SPMV. If combined with the SPMV from CBFPS, the accuracy of the SPMV can be greatly increased.

A more popular way to get the SPMV is to use a function (in most cases a second-order function) to model the SAD surface [7], [9]. If the matching errors of the best integer-pixel MV and its neighboring positions are known, the coefficients of the function can be solved. The position that corresponds to the smallest value in the SAD surface is then chosen as the SPMV.

Many functions can be used to model the SAD surface. Example second-order functions are listed as follows:

$$f(x,\ y) = c_1x^2 + c_2xy + c_3y^2 + c_4x + c_5y + c_6 \tag{2}$$

$$f(x,\ y) = c_1x^2 + c_2x + c_3y^2 + c_4y + c_5 \tag{3}$$

where x and y are coordinates of the surface, and $f(x,\ y)$ is the matching error (SAD) value. Normally, the best integer-pixel position is set to be located at (*0*, *0*), so its neighboring integer-pixel positions are at $(1, 0)$, $(-1, 0)$, $(0, 1)$, $(0, -1)$, and so on. As the number of model function coefficients increases, more integer-pixel neighboring SADs are needed.

In [7] and [9], (3) was used to determine one of the SPMVs, which used the best integer-pixel SAD and the SADs of its four diamond integer neighbors. Given these SAD values, the coefficients of (3) can be computed. The SPMV can then be calculated as follows:

$$SPMV = (x_p,\ y_p) = \arg\min_{x,\,y} f(x,\ y) = \left(\frac{-B}{2A},\ \frac{-D}{2C}\right) \tag{4}$$

where

$$\begin{cases} A = (I + J)/2 \\ B = (I - J)/2 \\ C = (K + L)/2 \\ D = (K - L)/2 \end{cases} \begin{cases} I = f(1,\ 0) - f(0,\ 0) \\ J = f(-1,\ 0) - f(0,\ 0) \\ K = f(0,\ 1) - f(0,\ 0) \\ L = f(0,\ -1) - f(0,\ 0) \end{cases}$$

$$\text{and} \begin{cases} f(0,\ 0) = SAD(0,\ 0) = c_5 \\ f(1,\ 0) = c_1 + c_2 + c_5 \\ f(-1,\ 0) = c_1 - c_2 + c_5 \\ f(0,\ 1) = c_3 + c_4 + c_5 \\ f(0,\ -1) = c_3 - c_4 + c_5. \end{cases}$$

If $(x_p,\ y_p)$ is a fractional vector, its components are quantized into quarter-pixel units.

Furthermore, Xu *et al.* [17] proposed to use early termination to further reduce the search points from the CBFPS method.

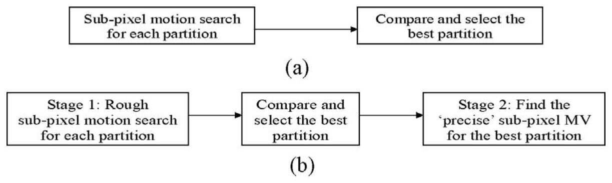

Sub-pixel motion search for each partition
Compare and select the best partition
(a)
Stage 1: Rough sub-pixel motion search for each partition
Compare and select the best partition
Stage 2: Find the 'precise' sub-pixel MV for the best partition
(b)

Fig. 1. Fast sub-pixel ME approaches. (a) Process for previous fast sub-pixel ME. (b) Proposed fast sub-pixel ME process.

III. Analysis on Reducing Sub-Pixel Search Points with Multiple Partitions

As shown in Section II, most previous fast sub-pixel ME methods reduce the number of search points by only searching the reduced area around the SPMV. For H.264/AVC multiple partition sizes, they attempt to find the "best" SPMV (with the smallest SAD) for each partition before the partition selection, as shown in Fig. 1(a).

However, in practice, only the best partition of the MB needs precise SPMVs. The MVs of other partitions are only used for the inter-mode selection. They are no longer useful after the best partition is selected. If a sub-pixel SAD is good enough to select the best partition, there is no need to search for more precise sub-pixel points in the first stage.

Therefore, if only a "rough" sub-pixel motion search is performed for each partition (the resulting MV does not necessarily have the smallest SAD), and a "precise" SPMV is determined only for the best partition selected, then the number of search points for the non-best partitions can be reduced greatly. As shown in Fig. 1(b), the purpose of the first stage ME is to obtain a rough sub-pixel SAD which is close to the best SAD. The integer-pixel SAD surface information can be used to decide whether the sub-pixel SAD is close to the best one or not. Based on the above discussion, we propose a new rough-strategy-based fast sub-pixel motion estimation algorithm (RFSME) described in detail in the next section.

IV. Fast Sub-Pixel ME Algorithm

The entire process of the proposed RFSME algorithm can be described in Fig. 2. In our algorithm, instead of using only the SAD to model the surface, we use COST [3], [10] as the ME matching cost in the rest of this letter. The COST [3], [10] is defined as follows:

$$COST = SAD + \lambda_{MOTION} \cdot R(MV) \tag{5}$$

where $R(MV)$ is the number of bits to code the MV and λ_{MOTION} is the Lagrange multiplier [11]. λ_{MOTION} is introduced to balance the importance between SAD and $R(MV)$. Note that COST can be viewed as a prediction of the total bits for coding both the matching error (i.e., SAD) and its side information (i.e., MV).

In Step 1, the difference between the best COST of the integer position and the two averaged COSTs of its four neighboring integer positions (the averaged COST of two vertical neighboring integer positions and the averaged COST of two horizontal neighboring integer positions) are checked. If the difference is small, it means that the COST surface is quite flat, and the best integer COST is close to the optimal sub-pixel COST (and, therefore, is good enough to estimate the best sub-pixel COST). In this case, the sub-pixel ME is skipped for the current partition. The best COST of the integer position is used in the partition selection in Step 4. The rule for deciding the COST surface flatness is shown as follows:

$$COST\_Surface = \begin{cases} Not\_Flat, & \text{if any of (a), (b), (c) is true} \\ Flat, & \text{otherwise} \end{cases} \tag{6}$$

where the conditions (a), (b), and (c) are as follows:

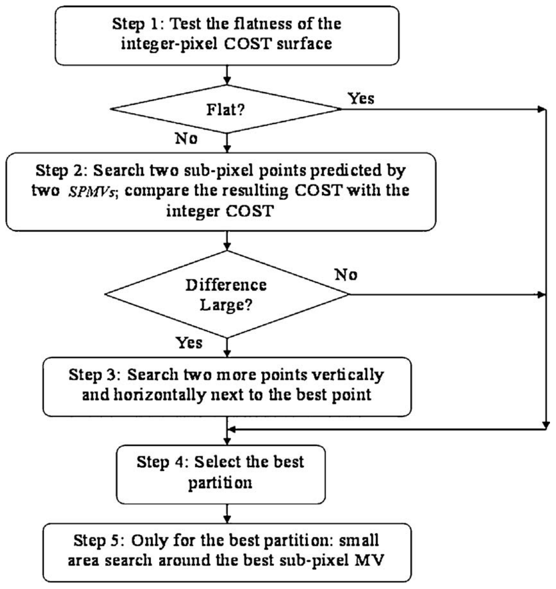
(a) $avg\_COST_{vertical} > r_F \cdot COST_{full}$ or $avg\_COST_{horizontal} > r_F \cdot COST_{full}$

(b) if $blocktype(i)$ $\min(|COST_{full} - avg\_COST_{vertical}|, |COST_{full} - avg\_COST_{horizontal}|) > th_1$

(c) if $blocktype(ii)$ $\min(|COST_{full} - avg\_COST_{vertical}|, |COST_{full} - avg\_COST_{horizontal}|) > th_2$

where $COST_{full}$ is the best COST after full-pixel ME, $avg\_COST_{vertical}$ is the COST average of its two vertical full-pixel neighbors, and $avg\_COST_{horizontal}$ is the COST average of its two horizontal full-pixel neighbors. r_F is a ratio parameter to decide whether $avg\_COST_{vertical}$ or $avg\_COST_{horizontal}$ is close to $COST_{full}$. $blocktype(i)$ represents 8×8, 8×4, 4×8, and 4×4 partitions, and $blocktype(ii)$ represents 16×16, 16×8, and 8×16 partitions. th_1 and th_2 are two thresholds. In the experiment of this letter, th_1, th_2, and r_F are set to 10, 20, and 5/4, respectively. These values are selected based on the experimental statistics.

If the COST surface is not flat in Step 1, in Step 2, two SPMV prediction methods are used to get two SPMVs. The first SPMV is calculated by the CBFPS method discussed in Section II, i.e., (1). The second SPMV is calculated by the second-order surface model discussed in Section II. After



Step 1: Test the flatness of the integer-pixel COST surface
Flat?
Yes
No
Step 2: Search two sub-pixel points predicted by two SPMVs; compare the resulting COST with the integer COST
Difference Large?
No
Yes
Step 3: Search two more points vertically and horizontally next to the best point
Step 4: Select the best partition
Step 5: Only for the best partition: small area search around the best sub-pixel MV

Fig. 2. Proposed RFSME.

TABLE I
DISTRIBUTION OF ABSOLUTE DISTANCE BETWEEN THE BEST SUB-PIXEL $MV(x_1, y_1)$ AND $MV_{step}2(x_2, y_2)$

Sequence	$d \leq 0$ (%)	$d \leq 1$ (%)	$d \leq 2$ (%)
*News*_QCIF	88.14	98.46	99.73
*Foreman*_QCIF	70.26	89.09	94.9
*Mobile*_QCIF	76.63	95.37	99.36

Note: $d = |x_1 - x_2| + |y_1 - y_2|$ in quarter-pixel units.

these two points are searched, these two points together with the best integer point are compared and the point that has the smallest COST is selected, namely, $COST_{step2}$. The MV that corresponds to $COST_{step2}$ is defined as MV_{step2}.

Table I lists the distribution of absolute distance ($d = |x_1 - x_2| + |y_1 - y_2|$ between the best sub-pixel (x_1, y_1) MV and (x_2, y_2) MV_{step2} (the predicted MV corresponding to $COST_{step2}$). The test condition is the same as that described in Section V. It shows that MV_{step2} can provide a good prediction of the best SPMV. For example, we can see from Table I that more than 70% MV_{step2} is exactly the same as the best SPMV and more than 94% MV_{step2} is within two quarter-pixel distance from the best SPMV. Therefore, after Step 2, the assumption is made that MV_{step2} is close to the best SPMV (but $COST_{step2}$ is not necessarily close to the best sub-pixel COST). The absolute difference between $COST_{step2}$ and the best integer-pixel COST in Step 1 ($COST_{best\_full\_pixel}$) is checked, i.e., $D = |COST_{step2} - COST_{best\_full\_pixel}|$.

If D is small, this means that the COST does not decrease much between $COST_{step2}$ and the best integer-pixel COST, and that $COST_{step2}$ is already close to the best sub-pixel COST and is good enough for the mode selection. In this case, $COST_{step2}$ is used in the partition selection in Step 4. The rule for deciding whether D is small or not can be described as follows:

$$D \quad \text{is} \quad \begin{cases} \text{Large}, & \text{if any of (a), (b), (c) is true} \\ \text{Small}, & \text{otherwise} \end{cases} \tag{7}$$

where

(a) $avg\_COST_{vertical} > r_D \cdot COST_{step2}$ or

$avg\_COST_{horizontal} > r_D \cdot COST_{step2}$

(b) $blocktype(i),\ \ D > \frac{1}{2}th_1$

(c) $blocktype(ii),\ \ D > \frac{1}{2}th_2$

where $avg\_COST_{vertical}$ and $avg\_COST_{horizontal}$ are the same as in (5). r_D is a ratio parameter to decide whether $avg\_COST_{vertical}$ or $avg\_COST_{horizontal}$ is close to $COST_{step2}$. It is set to 1.5 in this letter.

If D is large, $COST_{step2}$ may not be close to the best sub-pixel COST [as shown in Fig. 3(a)]. In this case, the two points vertically and the two points horizontally next to MV_{step2} in quarter-pixel resolution will be checked. As shown in Fig. 3(b), the black point is MV_{step2}, the gray points are quarter-pixel neighbors of MV_{step2}, and the white points are integer neighboring points of MV_{step2}. In Step 3, two search points are selected as one point out of V_1 and V_2, and one point out of H_1 and H_2. A bilinear model as described below is used to select one of the neighboring points. As shown in Fig. 4(a), the slopes are first computed [based on (9)] between the two horizontal neighboring integer points (or the two vertical neighboring integer points) and the best sub-pixel point from Step 2 (the point by MV_{step2}). Then, the quarter-pixel neighboring point is selected corresponding to the slope with the *smaller* slope value, as shown in Fig. 4(b) and (8) as follows:

$$P_{step3}^{Horizontal} = \begin{cases} H_1, & \text{if } S_{H1} < S_{H2} \\ H_2, & \text{if } S_{H1} > S_{H2} \end{cases}$$

$$\text{and}\quad P_{step3}^{Vertical} = \begin{cases} V_1, & \text{if } S_{V1} < S_{V2} \\ V_2, & \text{if } S_{V1} > S_{V2} \end{cases} \qquad (8)$$

where

$$S_i = \left| \frac{COST_{\text{integer}\_i} - COST_{step2}}{Coord_{\text{integer}\_i} - Coord_{step2}} \right|,\ i = V_1, V_2, H_1, H_2 \qquad (9)$$

where $integer\_i$ represents the *closest* integer-pixel point in is direction (i.e., V_1 and V_2 for the vertical direction and H_1 and H_2 for the horizontal direction), and $Coord$ is the coordinate (in quarter-pixel resolution) of the points. The X-coordinate (horizontal direction) is used for H_1 and H_2, and the Y-coordinate (vertical direction) is used for V_1 and V_2.

After Steps 1–3, a COST value ($COST_{rough}$) can be obtained for each partition, which is close or equal to the best COST. The SPMV that corresponds to $COST_{rough}$ is denoted by MV_{rough}.

In Step 4, $COST_{rough}$ is used to select the best partition. In Step 5, a small area sub-pixel refinement is performed around MV_{rough}. In the proposed algorithm, the eight quarter-pixel neighbors around MV_{rough} are searched. Since Step 5 is performed only for the best partition selected, the average search points per partition is reduced compared to conventional fast sub-pixel search algorithms.

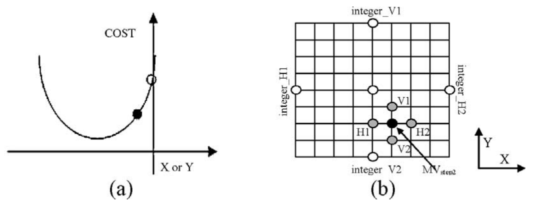


Fig. 3. (a) Example COST surface for $COST_{step2}$ not close to the best sub-pixel COST (the white and the black dots represent the positions for the best integer and MV_{step2}, respectively). (b) MV_{step2} and its quarter-pixel neighboring points.

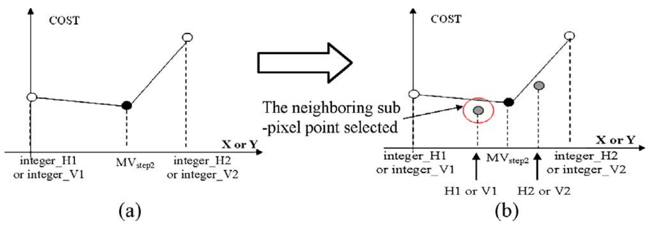


Fig. 4. Using the bilinear model to select neighboring search points (white points: integer pixel; black points: MV_{step2}; gray points: neighboring point selected). Note that in (a), the left slope is smaller than the right slope, therefore, in (b), the neighboring sub-pixel point on the left is selected.

It should be noted that the proposed RFSME algorithm is just one implementation of our idea described in Section III. Our method is general and it could also be implemented in other ways. For example, we can simply skip the sub-pixel search in the "rough" search step and directly use the best full-pixel searching results to select the partition, and then perform the "precise" search for the best partition. This can be viewed as a simplified version or extension of the RFSME algorithm.

V. Experimental Results

We implemented our proposed algorithm on the H.264/AVC reference software JM [5]. In the experiments, each test sequence of 100 frames is coded. The picture coding type is IPPP..., and the frame rate is 30 f/s. The search range is 16 for QCIF and 32 for CIF and standard definition (SD). The number of reference frames is 1. Full search is used for the integer pixel ME in our experiment [5]. It should be noted that our algorithm is general and various other integer pixel ME algorithms can also be easily implemented, as will be discussed later. Six methods are compared for each sequence as follows.

1) JM reference method [5] (sub-pixel full search).
2) The method in [6] (CBFPS).
3) The method in [7] (FPME).
4) The method in [8] (PDFPS).
5) Use the best integer COST directly to select the partition and then use JM's method to perform the sub-pixel ME for the best partition (IE+SME-Proposed). As mentioned, this method can be viewed as an extension of our RFSME algorithm.
6) The proposed RFSME method (RFSME-proposed).

TABLE II
COMPARISON OF DIFFERENT ME METHODS

Sequence	Method	PSNR (dB)	BR (kb/s)	SP/PT
*Akiyo*_QCIF (176 × 144) QP = 24	Full Search	40.82	56.05	16
	CBFPS	40.8	57.01	6.02
	FPME	40.81	57.12	2.92
	PDFPS	40.79	57.26	3.30
	IE + SME-Proposed	40.77	60.97	0.41
	RFSME-proposed	*40.8*	*57.05*	*0.87*
*Mobile*_QCIF (176×144) QP = 28	Full Search	32.95	453.39	16
	CBFPS	32.95	453.90	7.02
	FPME	32.95	455.82	5.81
	PDFPS	32.95	457.17	5.72
	IE + SME-Proposed	32.92	484.43	0.51
	RFSME-proposed	*32.95*	*456.61*	*3.1*
*Football*_CIF (352 × 288) QP = 28	Full Search	36.03	1440.84	16
	CBFPS	36.01	1448.87	7.63
	FPME	36.00	1455.60	6.21
	PDFPS	36.01	1452.18	6.85
	IE+SME-Proposed	36.01	1473.46	1.13
	RFSME-proposed	*36.01*	*1451.55*	*3.13*
*Football*_CIF (352×288) QP = 18	Full Search	43.15	4456.43	16
	CBFPS	43.15	4459.07	7.96
	FPME	43.14	4472.68	6.46
	PDFPS	43.14	4469.79	7.08
	IE + SME-Proposed	43.14	4516.51	1.35
	RFSME-proposed	*43.14*	*4462.82*	*3.69*
*Mobile*_SD (720 × 576) QP = 28	Full Search	33.8	8228.28	16
	CBFPS	33.79	8253.38	7.12
	FPME	33.78	8289.28	6.22
	PDFPS	33.79	8302.22	6.10
	IE+SME-Proposed	33.76	8625.27	1.24
	RFSME-proposed	*33.79*	*8293.79*	*2.88*
*Flower*_SD (720×576) QP = 24	Full Search	37.95	8428.84	16
	CBFPS	37.95	8432.12	6.3
	FPME	37.94	8449.21	5.97
	PDFPS	37.95	8461.3	5.9
	IE + SME-Proposed	37.92	8631.19	0.93
	RFSME-proposed	*37.95*	*8431.03*	*2.39*

In Table II, the peak signal to noise ratio (PSNR), BR, and average search points (SP) per partition size (SP/PT) [3], [7] for each method are compared for sequences in different resolutions and with different quantization parameters (QPs). The rate-distortion (R-D) curves for some sequences in Table II are shown in Fig. 5(a) and (b). Furthermore, Fig. 5(c) and (d) shows the BR-SP/PT curves for different methods.

Several observations can be drawn from Table II and Fig. 5.

The previous methods (CBFPS, FPME, and PDFPS) can reduce the SP by reducing the search area around the SPMV. However, our proposed methods (IE+SME-proposed and RFSME-proposed) can further reduce more than half the SP compared to previous methods (CBFPS, FPME, and PDFPS) by only performing the "precise" search on the best partition.

The IE+SME-proposed method can reduce the most number of search points, but the performance decrease is also large for some sequences [e.g., for *Akiyo*_QCIF in Fig. 5(a)]. This implies that only using the best integer-pixel COST may not always be able to find the best partition mode suitably, and some sub-pixel motion search may be needed to help select the best mode. However, due to its smallest number of SPs, the IE+SME-proposed method can still be very useful in situations where computation complexity is a crucial factor and some quality degradation is tolerable.

The RFSME-proposed method has the best overall performance. Compared with FS and other previous methods (CBFPS, FPME, and PDFPS), the RFSME-proposed method has much smaller SP while keeping almost the same coding performance. Compared with the IE+SME-proposed method, the proposed method has obviously better coding performance. With the RFSME-proposed method, the SP per partition size can be reduced to less than three for most sequences. The SP can be further reduced and becomes close to that of the IE+SME-proposed method for low motion videos (e.g., *Akiyo*_QCIF).

When QP decreases, the SP-per-partition-size for most of the methods will slightly increase. This is because: 1) the recovered reference frames are more precise for smaller QPs (i.e., higher PSNR). Therefore, the chance for the MB to select a smaller partition size becomes higher, and 2) when the reference frames are more precise, the COST surface for the

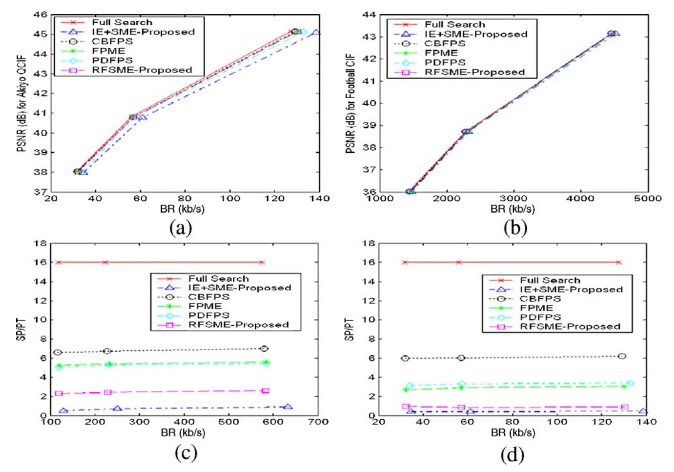


Fig. 5. R-D and BR-SP/PT curves comparisons for different methods. (a) R-D curve for *Akiyo*_QCIF. (b) R-D curve for *Football* CIF. (c) BR-SP/PT curve for *Foreman*_QCIF. (d) BR-SP/PT curve for *Akiyo*_QCIF.

interpolated sub-pixel locations may become more "complex," and it may take more steps to find the best sub-pixel location.

Besides the above observations, there are also other advantages of the proposed method. First, the proposed RFSME algorithm models the sub-pixel COST surface based on the four-neighboring integer COST values. Thus, the algorithm can be easily combined with most of the existing fast integer ME algorithms. Since most fast integer ME algorithms [4], [10], [14]–[16] (e.g., simplified hex search [14] and diamond search [15]) end the ME process by searching the four-neighboring points around the best integer point, using the four-neighbor COST information does not introduce any extra cost to the integer ME process. Furthermore, with the development of new video coding standards [such as high efficiency video coding (HEVC) and next generation video coding (NGVC)] [12], some existing sub-pixel ME methods may no longer work. For example, with the introduction of adaptive interpolation filter [13] in HEVC or NGVC, the second-order sub-pixel COST surface model may become unsuitable since the interpolation filter will adapt to the frame contents. This will greatly limit the usefulness of many fast sub-pixel methods [7], [9], which rely on this second-order model. Compared to these methods, our proposed methods can still work efficiently after some simple extensions. This is because: 1) the basic idea of our method is to reduce sub-pixel SP by performing "rough" search in the non-best partitions. As long as we can find some way to perform "rough" search, the proposed method can be easily applied to the new standards, and 2) there may be more partition sizes introduced in the future standards. In these cases, our proposed method can work even more efficiently by reducing the sub-pixel SP in the non-best partitions.

Table III shows another experiment for a multiple reference frame case. In this case, our algorithm first performs the "rough" search for all partitions on all reference frames and then performs the "precise" search only for the best partition on the best reference frame. From Table III, we can see that our algorithm can further reduce SP/PT by performing the "rough" search on those non-best reference frames.

TABLE III
RESULTS FOR *Football* CIF USING THREE REFERENCE FRAMES WITH QP = 28

	Full Search	CBFPS	FPME	PDFP	IE+SME	*RFSME*
PSNR (dB)	36.06	36.03	36.03	36.04	36.02	*36.03*
BR (kb/s)	1436.21	1442.47	1450.04	1449.15	1478.40	*1446.45*
SP/PT	16	7.41	6.12	6.76	0.39	*2.4*

VI. CONCLUSION

In this letter, a fast sub-pixel ME algorithm is proposed. The proposed algorithm performs a "rough" sub-pixel search before the partition selection, and performs a "precise" sub-pixel search for the best partition, thus can greatly reduce the sub-pixel SP. Experimental results showed that the proposed algorithm can reduce SP by more than half compared with the previous algorithms, with negligible performance decreases.

REFERENCES

[1] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuit Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.
[2] J. Zhang and Y. He, "Performance and complexity joint optimization for H.264 video coding," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 2. May 2003, pp. 888–891.
[3] W. Lin, D. M. Baylon, K. Panusopone, and M.-T. Sun, "Fast sub-pixel motion estimation and mode decision for H.264," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2008, pp. 3482–3485.
[4] Z. Zhou and M. T. Sun, "Fast macroblock inter mode decision and motion estimation for H. 264/MPEG-4 AVC," in *Proc. Int. Conf. Image Process.*, vol. 2. 2004, pp. 789–792.
[5] *JM 10.2* [Online]. Available: http://iphome.hhi.de/suehring/tml/download/old$_{-}$jm
[6] 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.
[7] J. F. Chang and J. J. Leou, "A quadratic prediction based fractional-pixel motion estimation algorithm for H.264," in *Proc. IEEE Int. Symp. Multimedia*, Dec. 2005, pp. 491–498.
[8] L. Yang, K. Yu, J. Li, and S. Li, "Prediction-based directional fractional pixel motion estimation for H.264 video coding," in *Proc. IEEE Int. Conf. Acou. Speech Signal Process.*, vol. 2. Mar. 2005, pp. 901–904.
[9] J. W. Suh and J. Jechang, "Fast sub-pixel motion estimation techniques having lower computational complexity," *IEEE Trans. Consumer Electron.*, vol. 50, no. 3, pp. 968–973, Aug. 2004.
[10] W. Lin, K. Panusopone, D. M. Baylon, and M.-T. Sun "A new class-based early termination method for fast motion estimation in video coding," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2009, pp. 625–628.
[11] T. Weigand, H. Schwarz, A. Joch, F. Kossentini, and G. Sullivan, "Rate-constrained coder control and comparison of video coding standards," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 688–703, Jul. 2003.
[12] *Documents of the Joint Collaborative Team on Video Coding* [Online]. Available: http://ftp3.itu.int/av-arch/jctvc-site
[13] Y. Vatis and J. Ostermann, "Adaptive interpolation filter for H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 2, pp. 179–192, Feb. 2009.
[14] X. Yi, J. Zhang, N. Ling, and W. Shang, "Improved and simplified fast motion estimation for JM," document JVT-P021, Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Poznan, Poland, Jul. 2005.
[15] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block matching motion estimation," *IEEE Trans. Image Process.*, vol. 9, no. 2, pp. 287–290, Feb. 2000.
[16] H.-Y. C. Tourapis and A. M. Tourapis, "Fast motion estimation within the H.264 codec," in *Proc. Int. Conf. Multimedia Expo*, vol. 3. 2003, pp. 517–520.
[17] X. Xu and Y. He, "Improvements on fast motion estimation strategy for H.264/AVC," *IEEE Trans. Circuit Syst. Video Technol.*, vol. 18, no. 3, pp. 285–293, Mar. 2008.