Signal Processing: *Image Communication* 31 (2015) 134–150

Contents lists available at ScienceDirect

Signal Processing: *Image Communication*

journal homepage: www.elsevier.com/locate/image

Computation-constrained dynamic search range control for real-time video encoder

Xianghu Ji, Huizhu Jia*, Jie Liu, Xiaodong Xie, Wen Gao

National Engineering Laboratory for Video Technology, Peking University, Beijing 100871, China

ARTICLE INFO

Article history:
Received 26 March 2014
Received in revised form
12 November 2014
Accepted 5 December 2014
Available online 29 December 2014

Keywords:
Video coding
Motion estimation
Dynamic search range
VLSI architecture

ABSTRACT

Search range (SR) is a key parameter on the search quality control for motion estimation (ME) of a real-time video encoder. Dynamic search range (DSR) is a commonly employed algorithm to reduce the computational complexity of ME in a video encoder. In this paper, we model an effective predicted motion vector (PMV) deviation metric to predict the relationship between SR and motion vector difference (MVD), according to the prediction differences of both temporal and spatial motions of neighboring blocks. In addition, a computation-constrained DSR (CDSR) control algorithm is proposed to manage the computational complexity while maximizing video coding quality in a real-time computational constrained scenario. The SR is dynamically determined by three factors: motion complexity, user-defined probability and computation budget. Compared to the conventional DSR algorithms, the proposed CDSR is an effective and quantifiable algorithm to allocate more computation budget to the blocks with high PMV deviations (such as motion object boundary), and less computation budget to the well-matched motion predicted blocks, while maintaining a constrained computation requirement. Experimental results show that the proposed CDSR control algorithm is an effective method to manage the computation consumption of the DSR algorithm while keeping similar rate–distortion (RD) performance. It can achieve about 0.1–0.3 dB average PSNR improvement when the computation consumption is restricted to a specific level as compared with its equivalent Fixed SR algorithm and can achieve about 50–90% computation savings when compared to the benchmarks. For ME with high performance Processing Element (PE) engine, the quality degradation caused by the proposed CDSR algorithm can be ignored.

1. Introduction

In the hybrid block-based video coding standards such as ITU-T H.26x [1,2] and ISO/IEC MPEG-x series [3], motion estimation (ME) plays a vital role in achieving high compression efficiency by removing temporal redundancy between successive video frames. Motion estimation (ME) is defined as the process of searching for an optimal motion vector (MV) that represents displacement of coordinates of the best matched block in a reference frame (past/future frame) for the block in the current frame. Motion vector prediction that predicts motion vectors by exploiting the correlation of MVs between spatial or temporal neighboring blocks [4,5] is a popular technique used in various video coding standards, and the predicted motion vector (PMV) is often adopted as the start point for many ME algorithms. For the window based ME algorithms, an area in the reference frame within the predefined search range around the start point (the

* Corresponding author at: Peking University, No. 5 Yiheyuan Road Haidian District, Beijing 100871, China. Tel.: +86 18910958581.
E-mail addresses: xhji@jdl.ac.cn (X. Ji), hzjia@pku.edu.cn (H. Jia), jliu@jdl.ac.cn (J. Liu), xdxie@pku.edu.cn (X. Xie), wgao@pku.edu.cn (W. Gao).

http://dx.doi.org/10.1016/j.image.2014.12.002
0923-5965/

collocated block or the PMV) is defined as the search window (SW). Then searches are conducted on the candidates within a SW of a reference frame. The ME calculates the matching costs of the candidates in the SW, and the candidate with the smallest matching cost is the best match. The most common criterion of the matching cost is the sum of absolute differences (SAD) between pixels of the current block and pixels of the reference candidate.

With the applications of more efficient coding techniques and the increasing requirement of higher video resolutions, the computation complexity of ME has been dramatically increased. Studies have been done and shown that more than 70% of the total encoding time and 90% of the total memory access are dedicated to the ME process [6]. Therefore, many fast algorithms have been proposed to reduce the computational complexity of ME. Among all the fast algorithms, search point reduction is a very straightforward and effective method [7] to accelerate the ME process. For one category of those search point reducing algorithms, such as the three-step search [8], the four-step search [9], the diamond search [10] and the hexagon-based search [11], search points reduction is achieved by applying inherent search patterns. Although those fast algorithms alleviate the problem of high computational complexity, they are usually not hardware friendly and suffer from the problems of performance degradation, unpredictable memory access or irregular data flow, etc. For the ME algorithms (full search, hierarchical search, etc.) in which searching points are mainly determined by the search window, SR reduction algorithms are another way to reduce the computational complexity of ME. Dynamic search range (DSR) algorithms have been proposed by researchers [12–21] recently. The proposed DSR algorithm dynamically adjusts the SR for each Macroblock (MB) according to the information given by previously encoded syntax element. Hong [12] proposed a DSR algorithm using the motion vector (MV) information of the adjacent and previously coded blocks. The predicted SR of the current MB was determined by the magnitude of the MVs of neighboring blocks. Xu [13] improved the algorithm proposed in [12] to make it more suitable for H.264/AVC [2]. Yamada [14] presented a two-stage SR modification algorithm that limits the SR by the MVs of neighboring blocks as well as the prediction error of the corresponding block in the reference frame. Zhang [15] determined the SR based on the frame complexity measured by the degree of motion activity at frame level. Shimizus [16] utilized the information of variable block size in H.264/AVC. He performed 16×16 mode or 8×8 mode at first and utilized the resulting motion vectors to reduce the SR for large block modes. Song [17] evaluated the relationship between SR and average MVs. The optimal SR for the current MB was determined based on the estimated average MVs and a pre-defined threshold. Ko [18] and Dai [19] used different zero mean 2-D distributions to model the probability density function (PDF) of MVD and the neighboring MVD information was adopted to estimate the distribution of the current MVD which determines the optimal SR. Lou [20] proposed a DSR algorithm by using both spatial and temporal MVPs as parameters to adaptively select the SR. Afterwards, the algorithm was further improved in [21] by using the variance of the MVP set which has been proved to be highly correlated with the SR value.

However, most of the previous works focus on the reduction of the SR while keeping similar rate–distortion (RD) performance, ignoring the constraints of implementation. In real-time video application, ME is mostly implemented as a dedicated hardware accelerator [22–24] in which processing is guaranteed to finish within strict time constraint to permit search for the best matching block. To achieve the highest degree of parallelism, pipelined architectures [25,26] are the most commonly used techniques for hardware accelerators. The pipeline is designed to maximize the system throughput while satisfying latency and resource constraints. In addition, in order to improve the pipeline performance, all the pipeline stages are expected to have a similar processing time. For a fixed time budget, searching through a larger SW usually requires a high level of parallelism and results in large silicon area consumption for the ME stage. Furthermore, the DSR algorithms usually give an unrestricted dynamic SR for each MB which leads to a large variation of the processing time for ME. The computing resources required to guarantee real-time processing must be significantly increased with the search range and frame size to an unacceptable degree. Meanwhile, it is difficult to estimate the scale of computing resources for the DSR ME accelerator in order to guarantee the real-time processing of the system. For this reason, the dynamic SR that leads to an unpredictable computing requirement for ME in conventional DSR algorithms is not desirable for the hardware implementation. Hence, practical hardware-friendly DSR algorithm should not only consider the trade-off between the SR reduction and RD performance, but also its implications on hardware implementation.

In this paper, in order to overcome the problems of the conventional DSR algorithms, we propose an effective PMV deviation metric to model the distribution of the relationship between SR and MVD according to the prediction differences of both temporal and spatial neighboring motions. Furthermore, a computation-constrained DSR control algorithm is proposed to manage the computational complexity while maximizing video coding quality in a real-time computational constrained scenario. The SR is determined by three factors: motion complexity, user-defined probability (the probability of the optimal MV to fall into the desired window) and computation budget. The computation-constrained DSR control algorithm can be easily adapted to industrial products of real-time video encoders, such as live TV broadcasting, video telephone, particularly the VLSI based solution.

The remainder of this paper is organized as follows. The proposed Dynamic Search Range (DSR) prediction algorithm is presented in Section 2. In addition, a novel PMV deviation metric is introduced to model the distribution of the relationship between SR and MVD. In Section 3, the computation-constrained DSR control algorithm is presented. Experimental results of comparisons with the traditional non-constrained DSR algorithms are shown in Section 4. Finally, some concluding remarks are given in Section 5.

2. Dynamic search range prediction

2.1. Modeling distribution of SR and MVD

In video sequences, motion vectors of the neighboring partitions are often highly correlated and each motion vector is predicted from vectors of nearby and previously coded partitions. PMV is derived based on previously calculated motion vectors. For the window based ME algorithms, searching is performed within a SW to find an optimal MV that gives the minimal pixel differences. The difference between the optimal MV and the predicted vector (MVD) is encoded and transmitted. The MVD distribution within a frame has been investigated [27,28], and zero mean Cauchy distribution with the scale parameter γ, having the 1-D probability density function (PDF)

$$f(x)=\frac{1}{\pi\gamma\left[1+(x/\gamma)^2\right]}, x\in \boldsymbol{R} \tag{1}$$

has been proven to be a better fit to the MVD distribution than the Laplacian or Gaussian distribution [19]. Here γ is a scale parameter which depends on the picture content. The assumption is that the x and y are independent Cauchy random variables for horizontal and vertical directions of MVD respectively with scale parameter of γ and zero mean. The 2-D joint PDF can be written as:

$$f(x,y)=\frac{1}{\pi^2\gamma^2\left[1+(x/\gamma)^2\right]\left[1+(y/\gamma)^2\right]}, x,y\in \boldsymbol{R} \tag{2}$$

Let

$$\begin{aligned}\Lambda(S_x,S_y,\gamma)&=f(|x|\le S_x,|y|\le S_y)\\&=\int_{-S_y}^{S_y}\int_{-S_x}^{S_x}f(x,y)dxdy\\&=\frac{4}{\pi^2}\tan^{-1}\left(\frac{S_x}{\gamma}\right)\tan^{-1}\left(\frac{S_y}{\gamma}\right)\end{aligned} \tag{3}$$

Here Λ is the probability of the optimal MV falling into the search window. S_x and S_y are the half ranges of the search window centered at the PMV in horizontal and vertical directions, respectively.

Let assume that a square search window is adopted for motion search. Let $S_x=S_y=S$, Eq. (3) can be simplified to

$$\Lambda(S,\gamma)=\frac{4}{\pi^2}\left[\tan^{-1}\left(\frac{S}{\gamma}\right)\right]^2 \tag{4}$$

then

$$S(\Lambda,\gamma)=\gamma\times\ \tan\left(\frac{\pi\sqrt{\Lambda}}{2}\right) \tag{5}$$

Therefore, given a model parameter γ, the optimal search range S with hitting probability Λ can be predicted by Eq. (5).

2.2. The proposed dynamic search range prediction algorithm

The SR can be modeled by Cauchy parameter γ and hitting probability Λ. If we can get a precise estimation of γ, the optimal SR for each search block with hitting probability Λ can be accurately predicted. As the Cauchy parameter γ determines the shape of the MVD distribution, its estimation γ is highly dependent on the accuracy of the MVP. If the MVP is accurate, the difference between the MVP and the optimal MV is small, which indicates a sharper MVD distribution. In order to get a precise estimation of γ for the MVD distribution, the confidence of the PMV for each searching block should be evaluated. As motion vector prediction is performed on the basis of the highly correlated motions, the prediction accuracy of PMV is determined by the prediction differences of temporal/spatial neighboring motion vectors. To model the deviation of the PMV, the concept of temporal and spatial motion prediction difference is introduced here.

As is shown in Fig. 1, block A, B, C and D are the neighborhood blocks in a current frame. Let assume that motion vectors of A, B, C and D are MV_1, MV_2, MV_3 and MV_4. Furthermore, the motion vector of each neighboring block is normalized according to its temporal distance:

$$\begin{aligned}MV_i^n&=\mathrm{Sign}(MV_i)\\&\times\left(\left(|MV_i|\times \mathrm{Dist_{cur}}\times\left(\frac{512}{Dist_i}\right)+256\right)\gg 9\right)\end{aligned} \tag{6}$$

where Dist is the temporal distance between current and reference frame in display order represented by number of frames. $\mathrm{Dist_{cur}}$ and Dist_i are the temporal distances for the current and neighboring blocks, respectively.

Then the spatial motion prediction difference δ_s can be calculated as

$$\delta_s=\sum_{i=0}^{N-1}\left[\omega_i\times\left(\left|MVX_i^n-PMVX\right|+\left|MVY_i^n-PMVY\right|\right)\right] \tag{7}$$

where N is the number of spatially available neighboring blocks. MVX_i^n and MVY_i^n are horizontal and vertical components of MV_i^n, respectively. PMVX and PMVY are motion vector components of PMV for the current block. ω_i is the weighting factor based upon the relative distance between the current block and its neighbor blocks.

To measure the temporal motion prediction difference, the temporal neighboring window and temporal motion vectors are introduced in this section. As shown in Fig. 2,

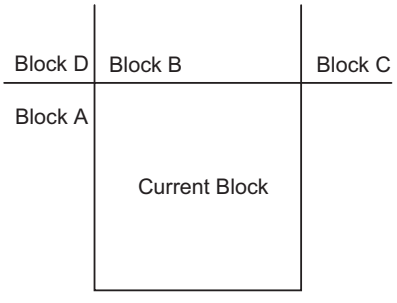

Fig. 1. Spatial neighborhood for current block.

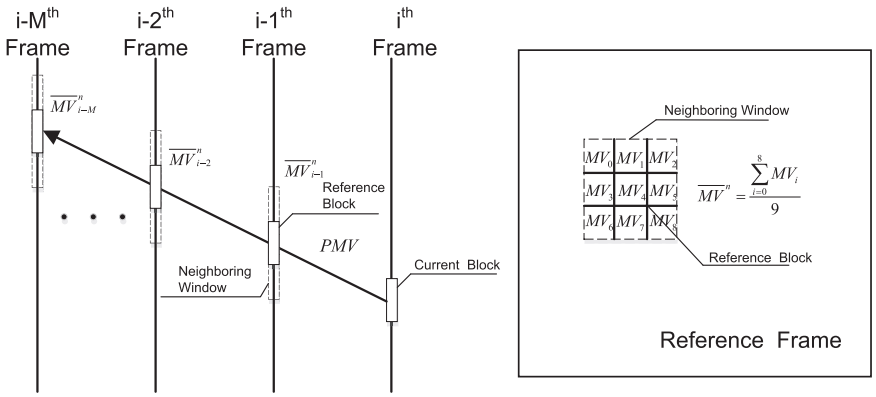

Fig. 2. Temporal neighborhood for current block.

the temporal neighboring window is the surrounding area centered by the reference block across multiple reference frames in time, temporal motion vectors $\overline{\mathrm{MV}}^n$ are the average of normalized motion vectors in a temporal neighboring window in space, and the temporal neighboring windows in different reference frames are predicted by the PMVs of the current block. Then the temporal motion prediction difference δ_t for current search block in ith frame can be calculated as

$$\delta_t = \sum_{j=i-1}^{i-M} \left[\tau_j \times \left(\left|\overline{\mathrm{MVX}}_j^n - \mathrm{PMVX}_{j-1}\right| + \left|\overline{\mathrm{MVY}}_j^n - \mathrm{PMVY}_{j-1}\right|\right)\right] \tag{8}$$

where M is the number of available temporal neighboring windows. $\overline{\mathrm{MVX}}_j^n$ and $\overline{\mathrm{MVY}}_j^n$ are horizontal and vertical components of normalized temporal motion vectors in the jth reference frame. PMVX_{j-1} and PMVY_{j-1} are motion vector components of derived PMV for the $(j-1)$th reference frame. The weighting factor τ_j is adjusted according to the time distance between reference frame and current frame. The temporal motion prediction difference δ_t represents the deviation of the predicted motion vector from a temporal neighboring window across multiple frames, and a small δ_t implies that the current block is either stationary in motion (e.g. scene background) or an internal part of a solid object (foreground).

Therefore, the PMV deviation metric θ is proposed according to the prediction differences of temporal and spatial neighboring motions and is given as

$$\theta = \frac{\delta_s + \delta_t + \overline{\delta}K}{N+M+K} \tag{9}$$

where δ_s and δ_t are the spatial and temporal motion prediction differences, respectively. N and M are the numbers of spatially and temporally available motion vectors, respectively. A prior-belief bayesian factor $\overline{\delta}K$ is adopted to model the few/zero available neighboring blocks that often occur at the sequence/frame boundary. K is the number of introduced "pseudo" neighboring blocks (extrapolated frame or boundary). $\overline{\delta}$ is defined as the average prediction difference of the previously coded blocks and is calculated as

$$\overline{\delta} = \sum_{i=1}^{N_{mbprev}} \left[\frac{\delta_s(i)+\delta_t(i)}{N(i)+M(i)}\right] / N_{mbprev} \tag{10}$$

with $\overline{\delta} = 1$, for the beginning of a sequence, where N_{mbprev} is the number of previously motion-estimated blocks. $\overline{\delta}$ is updated after the processing of each block. PMV deviation θ can be used to indicate the SR requirement of ME. If motion vectors are highly correlated in space and time, the estimated block will behave as a low PMV deviation which leads to a small SR for ME.

To model the relationship between PMV deviation θ and Cauchy parameter γ, over 60 sequences varying from CIF to 1080P are statistically analyzed. These sequences are coded by a H.264/AVC encoder (JM18.4) using full search (FS). IPPP picture structure, block size of 16, 2 reference frames and SAD as the cost function are used. By considering the balance between searching efficiency and coding performance, the SR is set to 32 for all sequences. We calculate the statistics from these data, including θ and the optimal MVDs. The MVDs are categorized on the basis of PMV deviation θ. The hypothesized Probability Density Function (PDF) for each MVD cluster is estimated by the method in [29]. A well-known 2-D KS test [30] is adopted to justify the correctness of hypothesized MVD PDFs. The statistic defined in Eq. (11) is used as the measure of similarity between the hypothesized PDF (the modeled MVDs) and the observed PDF (measured MVDs).

$$D = \max\left\{\left|\mathrm{CDF}_{\mathrm{modeled}}(s) - \mathrm{CDF}_{\mathrm{measured}}(s)\right|\right\}, s \in (-\mathrm{SR},\ \mathrm{SR}) \tag{11}$$

Here $\mathrm{CDF}_{\mathrm{modeled}}$ and $\mathrm{CDF}_{\mathrm{measured}}$ are cumulative PDFs of the model and the measured MVDs, respectively. SR is the size of the SW. Four representative θ values which are selected to cover the common range of PMV deviation are used for the 2-D KS test. The detailed 2-D KS test results on some representative sequences are shown in Table 1. For most of the tested sequences, the 2-D KS test usually has a small statistic D (0.01–0.20) and it implies that the hypothesized PDF matches well with the observed PDF.

PDFs for MVD distributions with different PMV deviations are shown in Fig. 3(a). We can find that the MVD

Table 1
Statistic of 2-D KS test on some representative sequences[a].

Sequences	Number of frames	$\theta=0.2$		$\theta=1.4$		$\theta=4.0$		$\theta=8.0$	
		$\hat{\lambda}$	D[b]	$\hat{\lambda}$	D	$\hat{\lambda}$	D	$\hat{\lambda}$	D
Foreman (CIF)	300	0.10	0.10	0.12	0.11	1.41	0.16	1.84	0.18
Stefan (CIF)	300	0.23	0.20	0.30	0.21	1.74	0.16	2.52	0.14
Coastguard (CIF)	300	0.02	0.15	0.04	0.09	1.29	0.09	N/A	N/A
City (720P)	300	0.02	0.05	0.03	0.04	0.58	0.17	1.55	0.15
Night (720P)	300	0.12	0.18	0.19	0.22	1.79	0.17	3.77	0.13
Spincalendar (720P)	60	0.11	0.14	0.18	0.19	1.28	0.18	2.32	0.14
BasketballDrive (1080P)	80	0.25	0.19	0.28	0.20	1.65	0.16	2.87	0.12
BlueSky (1080P)	80	0.07	0.04	0.07	0.04	0.6	0.20	0.90	0.18
ChristmasTree (1080P)	80	0.07	0.01	0.08	0.02	1.50	0.11	3.88	0.07
BQTerrace (1080P)	80	0.08	0.04	0.09	0.06	0.53	0.21	1.45	0.20
…									
Average[c]		0.07	0.08	0.10	0.12	1.30	0.14	2.37	0.09

[a] Only ten representative sequences are shown here.
[b] SR is set as 32.
[c] The averages are for the whole set of 60 tested frames.

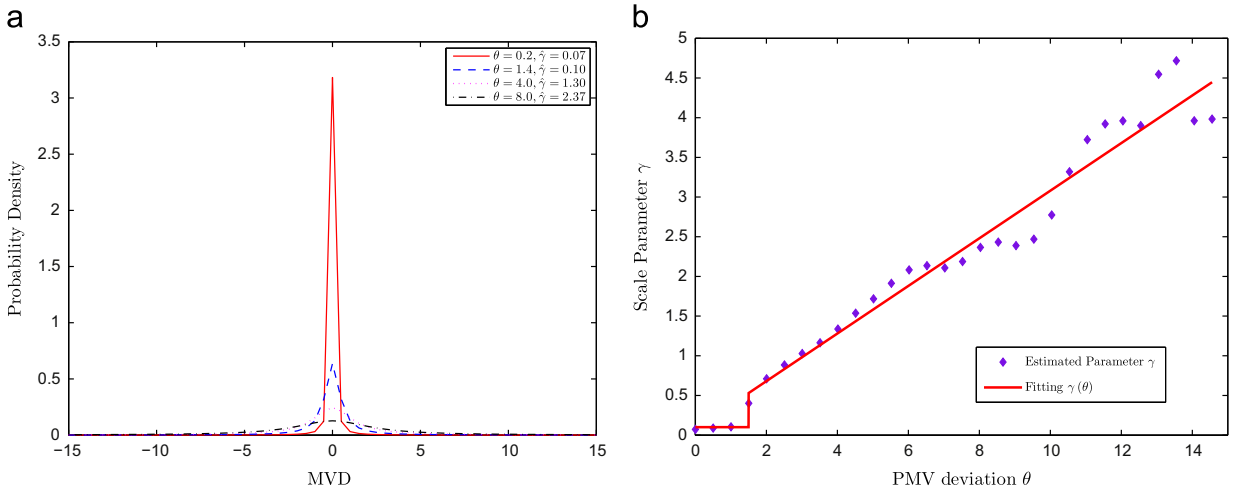

Fig. 3. (a) Probability density functions for MVD distributions with different PMV deviations. (b) Relationship between Cauchy parameter and PMV deviation.

distribution becomes flat with the increase of θ. Fig. 3(b) shows the relationship between Cauchy parameter and PMV deviation. The accuracy of PMV predicted by the motion vector prediction algorithm will increase with the increase of the motion correlation. However, the PMV is roughly estimated by using the motion vectors of the neighboring blocks. It is difficult to get an accurate PMV with high pixel precision (e.g. fractional pixel accuracy). Better prediction accuracy cannot be achieved with the decrease of the PMV deviation for the PMV at a specific pixel threshold (actually 1.5 pixel). The value of θ has little effect on the distribution of MVD and the Cauchy parameter γ keeps a constant value when $\theta<1.5$ (integer pixel corner), and when $\theta\geq 1.5$, the value of the Cauchy parameter γ grows linearly with the increase of PMV deviation θ. Hence, Cauchy parameter γ can be modeled as a piecewise function of the PMV deviation θ in the following form:

$$\gamma(\theta)=\begin{cases} c & ,\ \theta<1.5, \\ \xi_1\theta+\xi_2 & ,\ \theta\geq 1.5 \end{cases} \tag{12}$$

where c is a constant value. ξ_1 and ξ_2 are two coefficients of linear model. Regression is done by using the statistics of all tested sequences and the initial value of ξ_1, ξ_2 and c with the least regression error set to 0.3, 0.1 and 0.1, respectively. Therefore we can obtain the formulated relationship among search range S, hitting probability Λ and the estimated PMV deviation θ by substituting Eq. (12) into Eqs. (4) and (5):

$$\Lambda(S,\theta)=\frac{4}{\pi^2}\left\{\tan^{-1}\left[\frac{S}{\gamma(\theta)}\right]\right\}^2 \tag{13}$$

and

$$S(\Lambda,\theta)=\gamma(\theta)\times\ \tan\left(\frac{\pi\sqrt{\Lambda}}{2}\right) \tag{14}$$

Fig. 4 shows the relationship between search range S, hitting probability Λ and PMV deviation θ. We can see that the predicted search range varies greatly for different PMV deviations with a certain hitting probability, and very small search range is required for the blocks with low PMV deviations which are the majority in the video sequences. Therefore, by considering the PMV deviations and hitting probability, the search ranges for the motion estimated blocks can be dynamically predicted and the consequent complexity for the motion estimation can be dramatically decreased.

3. Computation-constrained DSR control

Since SR for motion estimation can be dynamically predicted by using the proposed DSR algorithm, the SR varies with the PMV deviation for each block. The variable SR usually implies different computing requirements for ME that may lead to a large variation in processing time. The unpredictable processing time for the DSR causes great limitation for real-time video applications where processing is desired to finish within strict time constraint. In this Section, the computation-constrained DSR control algorithm depending on the PMV deviation is proposed to manage the computational complexity while maximizing video coding quality in a real-time computational constrained scenario.

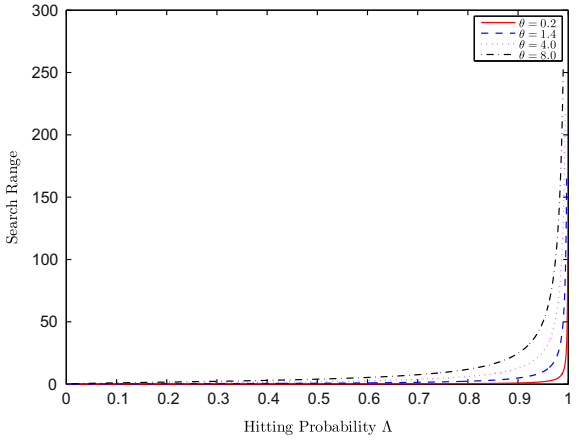

Fig. 4. Relationship among search range, hitting probability and PMV deviation.

3.1. Basic concept and models

3.1.1. Basic processing unit

Practical hardware architectures for real-time video applications [25,26] are often designed to work using a specific parallel granularity (typically MB level, slice level or frame level). To guarantee real-time processing, video processing algorithms are partitioned into several tasks and mapped into a specific parallel structure based on their computing requirements. In order to model the parallel granularity adopted in implementation, we introduce the concept of basic processing unit. Suppose that a frame is composed of N_{mbpic} MBs. A basic processing unit (BPU) is defined to be a group of contiguous MBs which is composed of N_{mbunit} MBs where N_{mbunit} is a fraction of N_{mbpic}. The number of BPUs in one frame N_{unit} can be given by:

$$N_{unit}=\frac{N_{mbpic}}{N_{mbunit}} \tag{15}$$

Basic processing unit can be a MB, a slice, or a frame. Computing resources are shared among all MBs in the same BPU. The BPUs in video sequence are illustrated in Fig. 5.

3.1.2. Motion estimation computation model

A typical motion estimation implementation usually has a limited computing capability for practical video applications. We can approximately model the computation capacity according to its implementation platform (CPU, DSP, ASIC etc.). Suppose computation budget T represents the computing capability (e.g. fixed number of instructions) that the computational core can provide per unit time. For software implementation T can be defined as follows:

$$T\approx\alpha\times \text{DMIPS} \tag{16}$$

DMIPS is dhrystone million instructions executed per second, and it represents the performance of a general-purpose processor (CPU). Parameter α is the proportion of the computational complexity of the ME task in the whole system. T denotes the number of instructions that the CPU

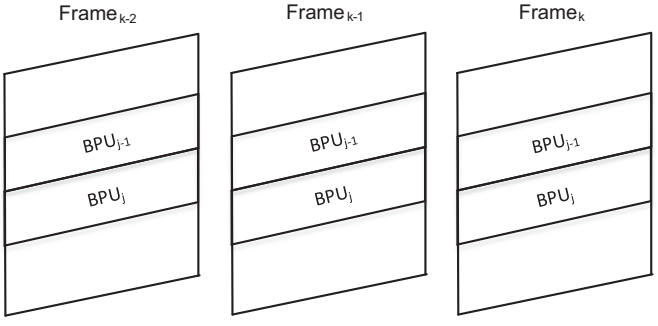

Fig. 5. Definition of basic processing unit (BPU).

Table 2
Computational complexity model of commonly used ME algorithms.

Motion estimation algorithms	Computational complexity model
Full search (FS)	$\Psi_{FS}(S) \approx \mathrm{MB}_{size} \times 4S^2 \times \varepsilon_1 + \varepsilon_2$
Three step search (TSS)	$\Psi_{TSS}(S) \approx \sum_{mode=8\times8}^{16\times16} \left\{ \mathrm{MB}_{size} \times 16S\left[1-\left(\frac{1}{2}\right)^{\left(\log_2^S+1\right)}\right] \times \varepsilon_1 \right\} + \varepsilon_2$
Multi-resolutions (MMEA)	$\Psi_{MMEA}(S) \approx \mathrm{MB}_{size} \times \left(\frac{S^2}{64} + 4S_{L1}^2 + 4S_{L0}^2\right) \times \varepsilon_1 + \varepsilon_2$

can provide for the ME task per unit time. For the VLSI implementation T can be given by:

$$T \approx \mathrm{Freq} \times N_{\mathrm{PE}} \tag{17}$$

where Freq is the system clock frequency. N_{PE} is the number of processing elements (PE) in the ME architecture. Suppose the input video sequence has the frame rate of R frames per second. The computation budget T_{pic} for performing ME for one frame can be given by:

$$T_{pic} = \frac{T}{R \times N_{ref}} \tag{18}$$

where N_{ref} is the number of reference frames. As the parallel granularity for the ME architecture is in the unit of BPU, the computation budget can only be shared within one BPU and this cannot be shared among different BPUs. Assume that each BPU in one frame has the same size and the time consumption switching between BPUs is ignored. Then, the computation budget T_{unit} for each BPU is obtained as follows:

$$T_{unit} = \frac{N_{mbunit} \times T_{pic}}{N_{mbpic}} \tag{19}$$

For the ME algorithms in which searching points are mainly determined by the search window, the computational complexity is strongly related to the SR. We formulate the computational complexity of commonly used ME algorithms as a function of SR and we list the results in Table 2, where S is the search range adopted in the specific ME. MB_{size} is the number of pixels in a MB with typical value of 16×16. ε_1 denotes the average computation budget consumption (number of instructions or clock cycles) per pixel. ε_2 is the computing overhead of data preparation (original and reference pixels fetch, search window load and pipeline setup, etc.) and finishing (motion vectors and costs output, etc.). Therefore, the computation budget constraint for ME can be given as

$$\sum_{j=1}^{N_{mbunit}} \Psi\left[S_{i,j}(k)\right] \le T_{unit} \tag{20}$$

where $S_{i,j}(k)$ is the search range for the ith MB in the jth BPU in the kth frame. Ψ is the computational complexity model for the adopted ME algorithm. To guarantee real-time processing, the computing requirement of ME for each BPU should be no larger than T_{unit}.

3.2. Proposed computation-constrained DSR control algorithm

Since computation capacity is limited for the real-time applications, the goal of our proposed computation-constrained DSR control algorithm is to minimize the overall performance degradation for ME within the constraints of the given computing resources. The problem is formulated as follows.

Let $\boldsymbol{S} = \{1, 2, \ldots, 1024\}$ be the set of search range values for motion estimation in a video encoder. Find $\boldsymbol{s}^* = \left\{S_{1,j}(k), \ldots, S_{N_{mbunit}j}(k)\right\}^T$, with $S_{i,j}(k) \in \boldsymbol{S}$ for $i = 1, 2, \ldots, N_{mbunit}$, where N_{mbunit} is the total number of MBs in the BPU, such that the hitting probability degradation with limited computation capability is minimized, i.e.

$$\boldsymbol{s}^* = \underset{\left(S_{1,j}(k),\ldots,S_{N_{mbunit}j}(k)\right)}{\arg\min} \sum_{i=1}^{N_{mbunit}} \left|\Lambda\left[S_{i,j}(k), \theta_i\right] - \Lambda_{TH}\right| \tag{21}$$

subject to the constraints

$$\begin{cases} \sum_{i=1}^{N_{mbunit}} \Psi\left[S_{i,j}(k)\right] \le T_{unit} \\ S_{i,j}(k) \le \gamma\left(\theta_{i,j}\right) \times \tan\left(\frac{2\sqrt{\Lambda_{TH}}}{\pi}\right) \end{cases}$$

where $\Lambda\left[S_{i,j}(k), \theta_i\right]$ is the hitting probability for the ith MB in the jth BPU in the kth frame with search range $S_{i,j}(k)$ and PMV deviation $\theta_{i,j}$. Λ_{TH} is the user defined hitting probability threshold which determines the performance of ME and T_{unit} is the computation budget for the current BPU. The formulated global optimization is to find a set of search ranges $\boldsymbol{s}^*$ by minimizing the overall hitting probability degradation (the highest coding efficiency) for the whole BPU. Two types of constraints are defined in Eq. (21) for global optimization:

(1) The real time constraint: not using more than the available computation budget for that BPU to guarantee overall real time operation.
(2) The performance constraint: the SR should be less than the derived user-defined threshold to guarantee the desired or better optimal MV hitting probability.

For real-time applications, it is impossible to obtain the global optimal solution to Eq. (21), due to the unavailability of future block information. However, it is possible to get a suboptimal solution based on the available information of the previously encoded blocks. Particularly, instead of explicitly minimizing the hitting probability degradation among the MBs, we address this problem by exploring a global computation budget allocation model developed to characterize the relationship between the allocated computation budget and the proposed PMV deviations.

The basic idea of our proposed computation-constrained DSR control algorithm is to develop an effective and quantifiable way to allocate more computation budget to the blocks with high PMV deviations (such as moving object boundary), and less computation budget to the well-matched motion

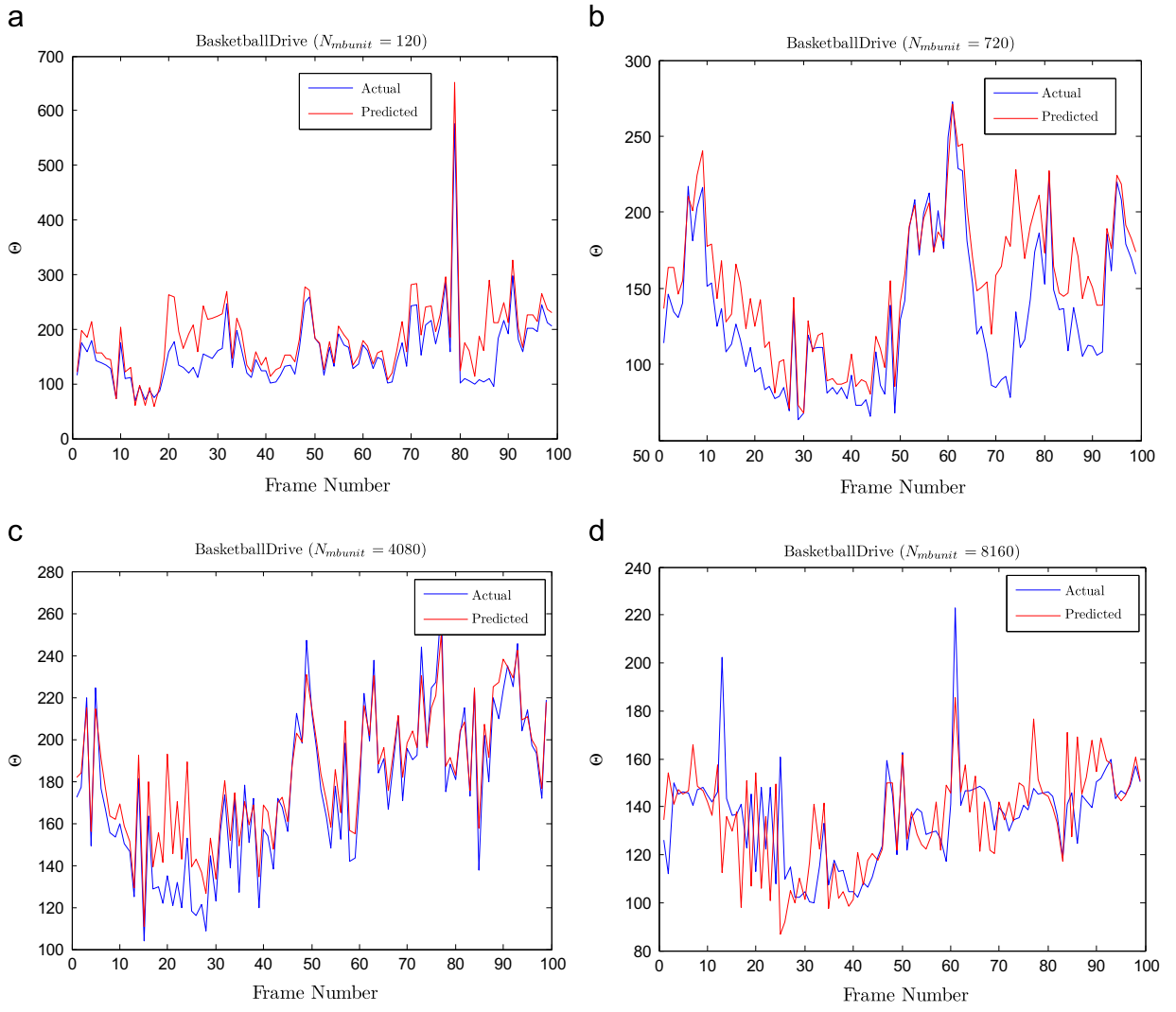

Fig. 6. Comparison of predicted and actual mean square of PMV deviations. (a) *BasketballDrive* with $N_{mbunit} = 120$. (b) *BasketballDrive* with $N_{mbunit} = 720$. (c) *BasketballDrive* with $N_{mbunit} = 4080$. (d) *BasketballDrive* with $N_{mbunit} = 8160$.

predicted blocks (internal MB of a moving object or the background), meanwhile maintaining a limited computing requirement, such that real-time processing is guaranteed.

3.2.1. A linear model for computational complexity prediction

Based on the computational complexity model given in Table 2, for most of the ME algorithms, the computational complexity Ψ is approximately proportional to the square of the search range. Moreover, as discussed in Section 2, the predicted search range is proportional to the PMV deviation according to Eqs. (12) and (14). Hence, we can derive that, the computational complexity Ψ is approximately proportional to the square of the PMV deviation θ.

$$\Psi \propto S^2 \propto \theta^2 \tag{22}$$

Let $\Theta_j(k)$ be the mean square of the PMV deviations $\theta_{*j}(k)$ of the jth BPU in the kth frame. Then, it can be easily derived that

$$\sum_{i=1}^{N_{mbunit}} \Psi[S_{i,j}(k)]/N_{mbunit} \propto \Theta_j(k) \tag{23}$$

where

$$\Theta_j(k) = \sum_{i=1}^{N_{mbunit}} \theta_{i,j}^2(k)/N_{mbunit}$$

Since computation budget consumption is directly related to the computational complexity, to optimally allocate the computation budget for each MB, the computational complexity should be first accurately determined by using the θ^2 of the current MB and the Θ of the whole BPU. However, the mean square of PMV deviations Θ for the current BPU is only available after processing the last MB in the BPU. It is known that motion usually behaves with high correlation between successive video frames. It

is possible to predict the Θ of the whole BPU from its neighboring frames. If some information that can be used for predicting Θ could be collected in previously coded frames, then the problem could be solved. According to the above analysis, a simple linear prediction model is proposed to predict the mean square of the PMV deviations for the current BPU:

$$\hat{\Theta}_j(k) = \lambda_1 \Theta_j(k-1) + \lambda_2 \tag{24}$$

where $\Theta_j(k-1)$ denotes the actual mean square of the PMV deviation of the jth BPU in the $(k-1)$th frame and $\hat{\Theta}_j(k)$ denotes the predicted mean square of the PMV deviations of the jth BPU in the current kth frame. λ_1 and λ_2 are two coefficients of the linear prediction model. The initial value of λ_1 and λ_2 are set to 1 and 0, respectively. They are updated after the processing of every BPU.

Fig. 6 shows the comparison of predicted and actual mean square of the PMV deviations for different sizes of BPU, for which I frames are excluded. We can see that the proposed linear prediction model with different sizes of BPU basically follows the trend of the actual curve, especially for frames with consistent motions. Although the linear prediction model could efficiently predict the actual change of Θ, for some sequences with fine texture details or irregular motion, this linear prediction model may not work as expected. However, it should be noted that the proposed linear prediction is just one $\Theta_j(k)$ prediction method. There are many other more accurate choices for the $\Theta_j(k)$ prediction. For example, look-ahead pre-processors that can collect the motion and texture information for the current frame before the coding process have been integrated into many practical video encoders recently [31,32]. A set of rough motion information for the current BPU can be obtained before DSR, and those rough motion vectors can be used for the current $\Theta_j(k)$ prediction.

3.2.2. Computation-constrained DSR control

To perform optimized computation budget allocation among MBs in a BPU, a computation budget tracking model based on the concept of BPU and ME computation model is proposed. Let $T_{i,j}(k)$ denote the number of remaining computation budget for the remaining $(N_{mbunit}-i)$ MBs in the jth BPU, and is computed as

$$T_{i+1,j}(k) = T_{i,j}(k) - t_{i,j}(k) \tag{25}$$

with

$$T_{1,j}(k) = T_{unit}$$

where $t_{i,j}(k)$ is the actual computation consumption for the ith MB in the jth BPU, and T_{unit} is the total computation budget for the current BPU.

Using the proposed computation budget tracking model (Eq. (25)) and the relationship between computational complexity and PMV deviation (Eq. (22)), the computation budget $\hat{t}_{i,j}(k)$ allocated for the ith MB in the jth BPU are determined based on the square of PMV deviations, the average computational complexity and the available remaining computation budget, and can be calculated by

$$\hat{t}_{i,j}(k) = \min\left\{\frac{\theta_{i,j}^2(k)}{\tilde{\Gamma}_{i,j}(k)} \times \frac{T_{i,j}(k)}{N_{mbunit}-i}, T_{i,j}(k)\right\} \tag{26}$$

In the above equation, $\tilde{\Gamma}_{i,j}(k)$ is the average computational complexity for the remaining MBs in the jth BPU. It is computed as follows:

$$\tilde{\Gamma}_{i+1,j}(k) = \tilde{\Gamma}_{i,j}(k) - \min\left\{\frac{\theta_{i,j}^2(k) - \hat{\Theta}_j(k)}{N_{mbunit}}, \tilde{\Gamma}_{i,j}(k) + \Delta\right\} \tag{27}$$

with

$$\tilde{\Gamma}_{1,j}(k) = \hat{\Theta}_j(k)$$

where $\hat{\Theta}_j(k)$ is the mean square of the PMV deviations predicted by the proposed linear prediction model (Eq. (24)), $\theta_{i,j}^2(k)$ is the square of PMV deviation for the ith MB in the jth BPU and Δ is a constant value introduced to avoid the model underflow, with a typical value of 1.

Depending on the allocated computation budget, the constraint for the computing resources can be given by

$$\varepsilon_2 \le \Psi[S_{i,j}(k)] \le \hat{t}_{i,j}(k) \tag{28}$$

$S_{i,j}(k)$ is the search range of ME for the ith MB in the jth BPU. ε_2 is the computing overhead of the ME core, and it is introduced to improve the searching efficiency.

Therefore, to maintain a bounded computing requirement for the DSR algorithm, the computation budget allocation is implemented by restricting the search range $S_{i,j}(k)$ determined by Eq. (14) for each MB, i.e.

$$S_{i,j}(k) = \min\left\{\max\left\{\Psi^{-1}[\varepsilon_2], \gamma[\theta_{i,j}(k)] \times \tan\left(\frac{\pi\sqrt{\Lambda_{TH}}}{2}\right)\right\}, \Psi^{-1}[\hat{t}_{i,j}(k)]\right\} \tag{29}$$

where $\theta_{i,j}(k)$ and Λ_{TH} are the PMV deviation and user defined hitting probability threshold for the current MB. $\gamma[\theta_{i,j}(k)] \times \tan(\pi\sqrt{\Lambda_{TH}}/2)$ is the dynamic search range predicted by the proposed DSR algorithm, respectively. $\Psi^{-1}[\hat{t}_{i,j}(k)]$ is the maximum search range for the ME engine with the allocated computation budget $\hat{t}_{i,j}(k)$, and it is adopted as the upper bound for the predicted search range to guarantee real-time processing. $\Psi^{-1}[\varepsilon_2]$ is set as the lower bound for the CDSR to improve the searching efficiency.

Since silicon area, memory bandwidth, data dependency and power consumption are four key factors for VLSI implementation, we evaluate the implementation complexity of our CDSR algorithm from four aspects:

(1) Silicon area. As is described in Sections 2 and 3, we can find that both $\theta_{i,j}(k)$ and $\hat{t}_{i,j}(k)$ behave with a limited value range. Therefore, for the hardware implementation, the results of $\gamma[\theta_{i,j}(k)] \times \tan(\pi\sqrt{\Lambda_{TH}}/2)$ and $\Psi^{-1}[\hat{t}_{i,j}(k)]$ can be kept on a predefined lookup table that can be stored on an on-chip RAM and 512 bytes are enough for the approximation precision. The lookup table can be configured by the micro control unit (MCU) at the beginning, and it will be updated after the changing of model parameters (Λ_{TH}, ξ_1 and ξ_2). Moreover,

Table 3
The proposed computation-constrained DSR control algorithm.

Step 1:	If it is the first inter frame or scene change frame in a sequence, initializing coefficients of the Cauchy parameter estimation model ($\xi_1=0.3, \xi_2=0.1, c=0.1$) and linear PMV deviation prediction model ($\lambda_1=1, \lambda_2=0$), set the value of the average prediction difference $\overline{\delta}$ to 1
Step 2:	Getting the total computation budget $T_{unit}=(N_{mbunit}\times T_{pic})/N_{mbpic}$ and user defined hitting probability threshold Λ_{TH} for the current BPU
Step 3:	Predict computational complexity $\hat{\Theta}_j(k)$ based on the linear prediction model and complexity of co-located BPU in the previous frame for the current BPU
Step 4:	Predicting the search range for the current MB in BPU. Step 4 is composed of the following 4 sub-steps
Step 4.1:	Normalize the motion vectors (MV_i) of neighboring blocks. Calculate the spatial motion prediction difference δ_s and temporal motion prediction difference δ_t by Eqs. (7) and (8), respectively. Compute the PMV deviation $\theta_{i,j}(k)$ for the current MB by Eq. (9)
Step 4.2:	*If* it is the first MB in the BPU, initialize the remaining computation budget $T_{1,j}(k)=T_{unit}$ and the average computational complexity $\tilde{\Gamma}_{1,j}(k)=\hat{\Theta}_j(k)$; *else*, updating the remaining computation budget $T_{i,j}(k)$ and computational complexity $\tilde{\Gamma}_{i,j}(k)$ by Eqs. (25) and (27)
Step 4.3:	Calculate the allocated computation budget $\hat{t}_{i,j}(k)$ by Eq. (26). Get the lower and upper bounds for the predicted search range
Step 4.4:	Compute the predicted search range $S_{i,j}(k)$ with the hitting probability threshold Λ_{TH} by Eq. (14), and to maintain a limited computing requirement for current BPU, $S_{i,j}(k)$ is clipped within the bound $\Psi^{-1}[\varepsilon_2], \Psi^{-1}[\hat{t}_{i,j}(k)]$ according Eq. (29)
Step 5:	Perform ME with the predicted search range $S_{i,j}(k)$ obtained in Step 4. Update the actual computation consumption $t_{i,j}(k)$ for the current MB and the average prediction difference $\overline{\delta}$. *If* the last MB of current BPU is reached, go to Step 6; *else*, go to Step 4
Step 6:	Update the Cauchy parameter estimation model parameters (ξ_1 and ξ_2) and linear PMV deviation prediction model parameters (λ_1 and λ_2) based on the actual computational complexity $\Theta_j(k)$ and statistics of the optimal motion vectors. *If* it is the last BPU in the frame, go to Step 7; *else*, go to Step 2
Step 7:	*If* the sequence ends, terminate the procedure; *else*, go to Step 1

it is worth noting that $\Psi^{-1}[\varepsilon_2]$ is a constant value for a specific ME implementation, and no extra circuits will be involved for $\Psi^{-1}[\varepsilon_2]$. After the calculation of $\theta_{i,j}(k)$ and $\hat{t}_{i,j}(k)$, the predicted computation-constrained search range $S_{i,j}(k)$ can be quickly calculated by querying the predefined on-chip lookup table. In addition, the linear computational complexity prediction can be divided into two parts handled by the hardware and software, respectively. The values of θ^2 are accumulated for each BPU by the hardware circuit. The accumulations will be read back by the MCU at the end of the frame and the linear prediction defined by Eq. (24) for each BPU can be calculated by the software. Hence, little silicon area overhead will be introduced for the implementation of our proposed CDSR algorithm.

(2) Memory bandwidth. Neighboring motion vectors needed for the calculation of the PMV deviation in the CDSR algorithm are the main contributing factors for the increase of system memory bandwidth. The spatial motion prediction difference δ_s is calculated from the motion vectors of spatially neighboring blocks. To reduce the bandwidth requirement, those spatially neighboring motion vectors can be stored on a line buffer. Assume that an integer motion vector occupies 20 b (10 b per component) in the memory. Then for 1080P@30fps video encoder, the size of the line buffer in the CDSR ME is (1920/16 MBs × 4 blocks + 4 blocks) × 20 b = 1210 B. Furthermore, those spatially neighboring motion vectors are the same as the vectors used in the MVP algorithm. Hence, the line buffer that stores the spatially neighboring motion vectors in the CDSR ME can be shared with the MVP module. Temporal neighboring windows that are used to calculate the temporal motion prediction difference δ_t can be loaded from external memory. Assume that the CSDR with a maximum of two temporal neighboring windows are adopted in the design. The memory bandwidth required for loading the temporal neighboring windows is 2 × 9 blocks × 20 b × 8160 MBs × 30 fps = 10 MBps. DDR2 is a commonly used external memory in VLSI design. It can provide high memory bandwidth, but in practice, the bandwidth utilization is low due to the random-access nature and read-modify-write dependencies of an application. According to the data given in the previous work [33], 80% is adopted as the DDR2 bandwidth utilization efficiency on average. For the 64-bit DDR2 working at 300 MHz, it can offer the average bandwidth of 3840 MBps. The extra external memory bandwidth requirement of the proposed CDSR is just 0.3% of the total system bandwidth. In addition, the bandwidth for loading the SW from the external memory is also a critical issue for the ASIC implementation when using a dynamic search range. To solve the bandwidth problem, the data reuse scheme should be specially designed for the DSR ME engine. The external memory bandwidth for the DSR ME engine can be reduced by applying some irregular data reuse schemes, such as cache based data reuse [34]. Meanwhile, bandwidth can also be reduced by maintaining a large SW in the ME engine and the reference pixels used by the processing element (PE) in DSR ME can be read directly from the large SW. The large SW can be updated by the traditional data reuse schemes, such as level C [35], level C+ [36], etc.

(3) Data dependency. The calculation of spatial motion prediction difference δ_s for the current MB is dependent on the final MVs of its spatial neighboring blocks which are not available until the optimal MB mode is determined and it involves some additional data dependencies in the ME process. The constraint of data dependency brought by the CDSR can be eliminated by some hardware oriented modifications (zigzag scan, data approximation, etc.)

which are the commonly used techniques for the MB pipelining architectures. Take zigzag scan [36] for example, several successive MBs in a horizontal direction will be processed before the vertical scan starts, and MB rows are processed alternately in the vertical scan. The former can guarantee that the data dependencies of neighboring MBs are satisfied. The performance of the proposed CDSR algorithm is influenced only for the MBs at the beginning and at the end of the frame. For high definition applications, the performance degradation can be ignored.

(4) Power consumption. Benefiting from the predictable processing time, the variable frequency technique can be easily integrated with the proposed CSDR algorithm. The system operating frequency can vary according to the computing requirement of ME. For the BPU (a MB, a slice, or a frame) that the predicted computing requirement is lower than the computation budget, operating frequency can be dynamically decreased to a specific level to save the power consumption.

Therefore, the proposed CDSR algorithm can be easily integrated into a VLSI-based ME architecture.

Our proposed CDSR control algorithm is described in detail in Table 3. The notations used in the proposed computation-constrained dynamic search range algorithm are summarized in Table 4.

With the dynamic search range prediction algorithm and computation budget allocation model, the proposed CDSR algorithm can allocate more computation budget to the block with high PMV deviation and less computation budget to the well-matched motion predicted block, while maintaining a limited computing requirements such that real-time processing is guaranteed.

4. Experimental results and discussions

The proposed computation-constrained DSR control algorithm is implemented on JM18.4 of H.264/AVC under conditions: *Profile/Level: 77/40, Reference frames: 2, Full Search ME, RC on* and *CABAC, IPPP* encoding structure. Computation consumption was theoretically calculated based on the previously introduced computational complexity model and on the SR.

To exhibit the advantages of the proposed algorithm, different types of ME Engines based on the 2D PE array [37] as shown in Table 5 are compared to in our experiment. The

Table 4
Notations used in the proposed computation-constrained DSR control algorithm.

Notation	Definition
N_{mbpic}	Total number of MBs in a frame
N_{mbunit}	Number of contiguous MBs in a BPU
γ	Cauchy parameter for MVD distribution
Λ_{TH}	Probability that an optimal MV falls into a search window
$\mathrm{Dist_{cur}}$,Dist_i	Reference distances for current and neighboring blocks
δ_s,δ_t	Spatial and temporal motion prediction differences for current block
$\overline{\delta}$	Average prediction difference for previously coded blocks
θ	PMV deviation metric for current block
ξ_1,ξ_2,c	Coefficients for the Cauchy parameter estimation model
T_{pic},T_{unit}	Computation budget for one frame and one BPU, respectively
Ψ	Computational complexity for a ME algorithm
ε_1,ε_2	Computation consumption per pixel and overhead of data preparation
$\Theta_j(k)$,$\hat{\Theta}_j(k)$	Actual and predicted mean square of PMV deviation
λ_1,λ_2	Two coefficients of linear prediction model
$T_{i,j}(k)$	Number of remaining computation budget for the remaining $(N_{mbunit}-i)$ MBs
$t_{i,j}(k)$,$\hat{t}_{i,j}(k)$	Actual and estimated computation consumptions for the ith MB in the$j$th BPU
$\tilde{\Gamma}_{i,j}(k)$	Average computational complexity for the remaining MBs in the jth BPU
$S_{i,j}(k)$	Predicted search range of ME for the ith MB in the jth BPU

Table 5
2D PE array based ME engines.

ME engine	N_{PE}	$(\varepsilon_1, \varepsilon_2)$	Working frequency(MHz)	Supported Video Spec.	Equivalent fixed SR
Standard	256	(1, 2,048)	2–10	CIF@30fps	[−5, 4]–[−11, 10]
			10–50	SD@30fps	[−6, 5]–[−10, 9]
High performance	1024	(1, 16,384)	25–100	720P@30fps	[−11, 10]–[−22, 21]
			75–256	1080P@30fps	[−13, 12]–[−23, 22]

ME Engines are configured with a different scale for the PE array and working frequency to satisfy the throughput requirements of different video specifications.

We defined four measures for evaluating the performance of CDSR, including average search range (ASR), worst computing utilization rate (WCR), average PSNR difference compared with its equivalent Fixed SR algorithm (ΔESR) and average PSNR difference compared with its computation-unlimited DSR algorithm (ΔUN). ASR denotes the average search range for the DSR ME in the whole video sequence. WCR denotes the worst-case computing utilization rate for the DSR algorithm implemented on the specific ME engine. It indicates the required computing resources for the DSR ME to guarantee the real-time operation. If the value of the WCR is higher than 100%, the real-time processing of the system is not guaranteed on the adopted ME engine. ΔESR and ΔUN are used to compare the coding performance difference between different DSR algorithms. We utilize the popular method proposed in [38] for calculating the average PSNR differences between R and D curves. And in order to evaluate the performance of the algorithm objectively, we choose four video sequences with different resolutions and different degree of movement for the performance evaluation. The proposed computation-constrained DSR algorithm was implemented based on the ME Engines listed in Table 5.

Due to the limitations of the system architecture and of the linear prediction model, the size of BPU has to be carefully chosen for the proposed CDSR algorithm before the evaluation. In Fig. 7, the SRs and PSNR curves of the proposed algorithm for the *BasketballDrive* sequence is shown with four BPU sizes: 120 (1 MB row), 720 (6 MB rows), 4080 (half frame) and 8160 (one frame).

We notice that the size of BPU influences the performance of the CDSR algorithm. There is an obvious PSNR degradation for the scenario with high motions (80–90th frame) when one MB row is adopted as the BPU size. Such observation accords with our adopted linear prediction model. As explained in the previous section, the computational complexity Θ for the current BPU is linearly predicted from its co-located BPUs in previous frames. If the size of the BPU is not large enough, such as smaller than one MB row, the prediction accuracy of the linear prediction model cannot be guaranteed for the large/irregular motion. Meanwhile, the range of computation budget allocation is very limited for the small BPU size and the accuracy for the computational complexity prediction will have great effect on the performance of CDSR. If finer grained parallelism is required for the system architecture, the performance degradation can be alleviated by adopting a more accurate computational complexity prediction method such as look-ahead pre-processing rather than the linear prediction model.

We compare the SRs and image quality degradation of different DSR algorithms and their equivalent fixed-SR in Figs. 8 and 9, respectively, where the size of BPU was set as six MB rows and the hitting probability Λ_{TH} was set as 95%. The computation-constrained DSR algorithm with unlimited computing resources ($T_{pic}=\infty$) and DSR algorithms proposed by Lou [21] and Song [17] were implemented for performance benchmarking. The "StME" and "HiME" denote the CDSR algorithm with standard and high performance ME engines, respectively. The "Unlimited" denotes the CDSR algorithm with unlimited computing resources. The "*FS@ESR*" represents the fixed-SR full search algorithm with a specific equivalent SR.

In Fig. 8, we can see that the proposed computation-constrained DSR algorithm maintains a very stable computation consumption when compared to the benchmarks which used dynamic search range prediction techniques but without a computing management scheme. The computing requirements of the Song's algorithm vary greatly with the video content. It shows large SRs for the *Stefan* and *Pingpong* sequences while maintaining very low SRs for the *Night* and *BasketballDrive* sequences. The

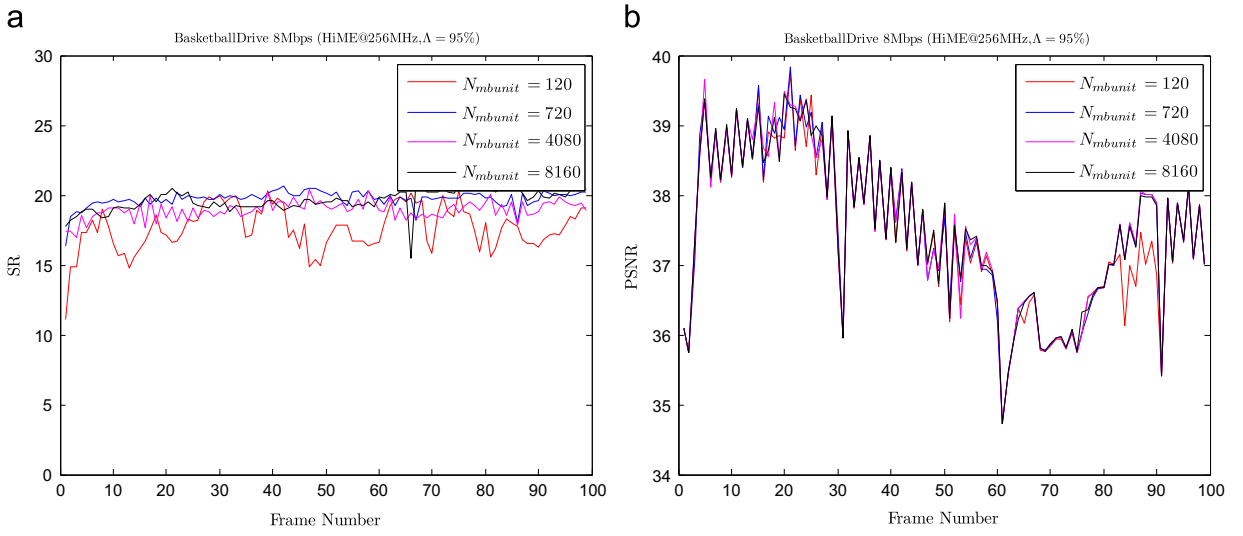

Fig. 7. Comparisons of CDSR with different sizes of BPU. (a) Search range comparisons for *BasketballDrive* 8 Mbps with HiME@256Mhz and $\Lambda_{TH}=95\%$. (b) PSNR curves for *BasketballDrive* 8 Mbps with HiME@256Mhz and $\Lambda_{TH}=95\%$.

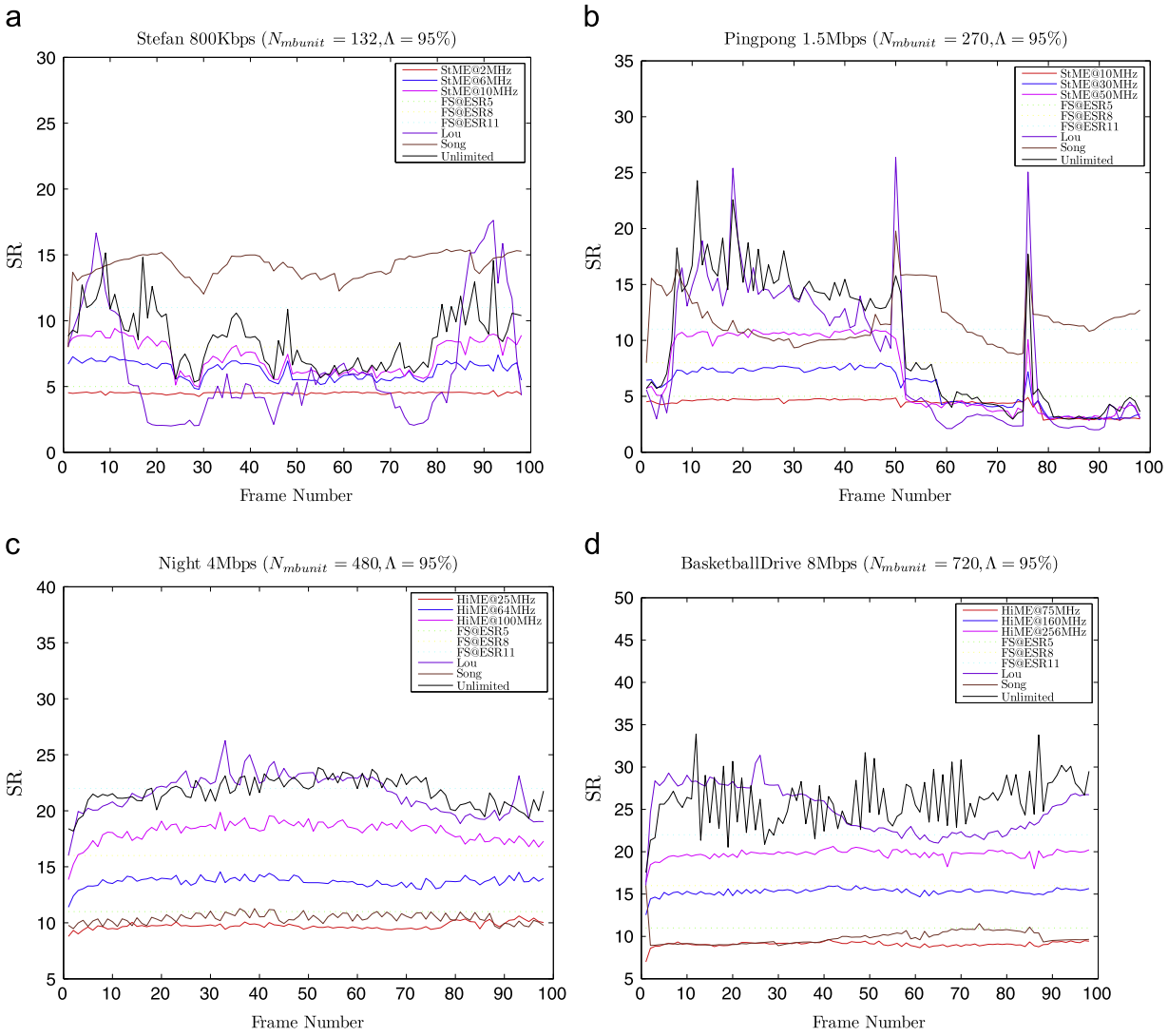

Fig. 8. Search range comparisons for different DSR algorithms. (a) *Stefan* 800 kbps with $N_{mbunit}=132$ and $\Lambda_{TH}=95\%$.(b) *Pingpong* 1.5 Mbps with $N_{mbunit}=270$ and $\Lambda_{TH}=95\%$. (c) *Night* 4 Mbps with $N_{mbunit}=480$ and $\Lambda_{TH}=95\%$. (d) *BasketballDrive* 8 Mbps with $N_{mbunit}=720$ and $\Lambda_{TH}=95\%$.

computing requirements of the Lou's algorithm basically follow the complexity of motions in the test sequences. However, the SR in Lou's algorithm is dramatically increased for the frames with various motions (e.g. *Pingpong* 50–55th frame). The SR values above the line of equivalent fixed SR indicate that the real-time processing is not guaranteed on the corresponding ME Engines for the benchmarks. The proposed CDSR algorithm allocates more computing resources to the fast moving objects in the scene, such as *Stefan* (90–100th frame) and *BasketballDrive* (80–90th frame). Fig. 9 shows that the proposed CDSR algorithm provides higher video quality compared with their equivalent fixed SR algorithm. *R–D* curves for different DSR algorithms are shown in Fig. 10. The video quality of the proposed algorithm improves with the increase of computation capacity, especially for the scenes with fast moving foregrounds. Moreover, the proposed CDSR with unlimited computing resources achieves the highest coding performance compared with the benchmarks. The superior coding performance originates from the more accurate model between SR and MVD, and the computation budget allocation algorithm. Song's algorithm shows very low search range for the *Night* and *BasketballDrive* sequences. However, the computation reduction is achieved at the expense of RD performance degradation (e.g. *BasketballDrive* 60–80th frame) compared with other DSRs.

Fig. 11 shows the visual quality results of the "*Stefan* (CIF)" sequence among different DSR algorithms. There are visible differences in some regions (e.g. the body boundary and the smooth background) of the decoded frame. Song's algorithm and the proposed CDSR with unlimited computing resources have the best visual quality. The proposed CDSR with StME@6Mhz perform better than its equivalent Fixed SR algorithm and has a similar visual quality compared with Lou's algorithms. The detailed experimental results are given in Table 6. From the observation of the

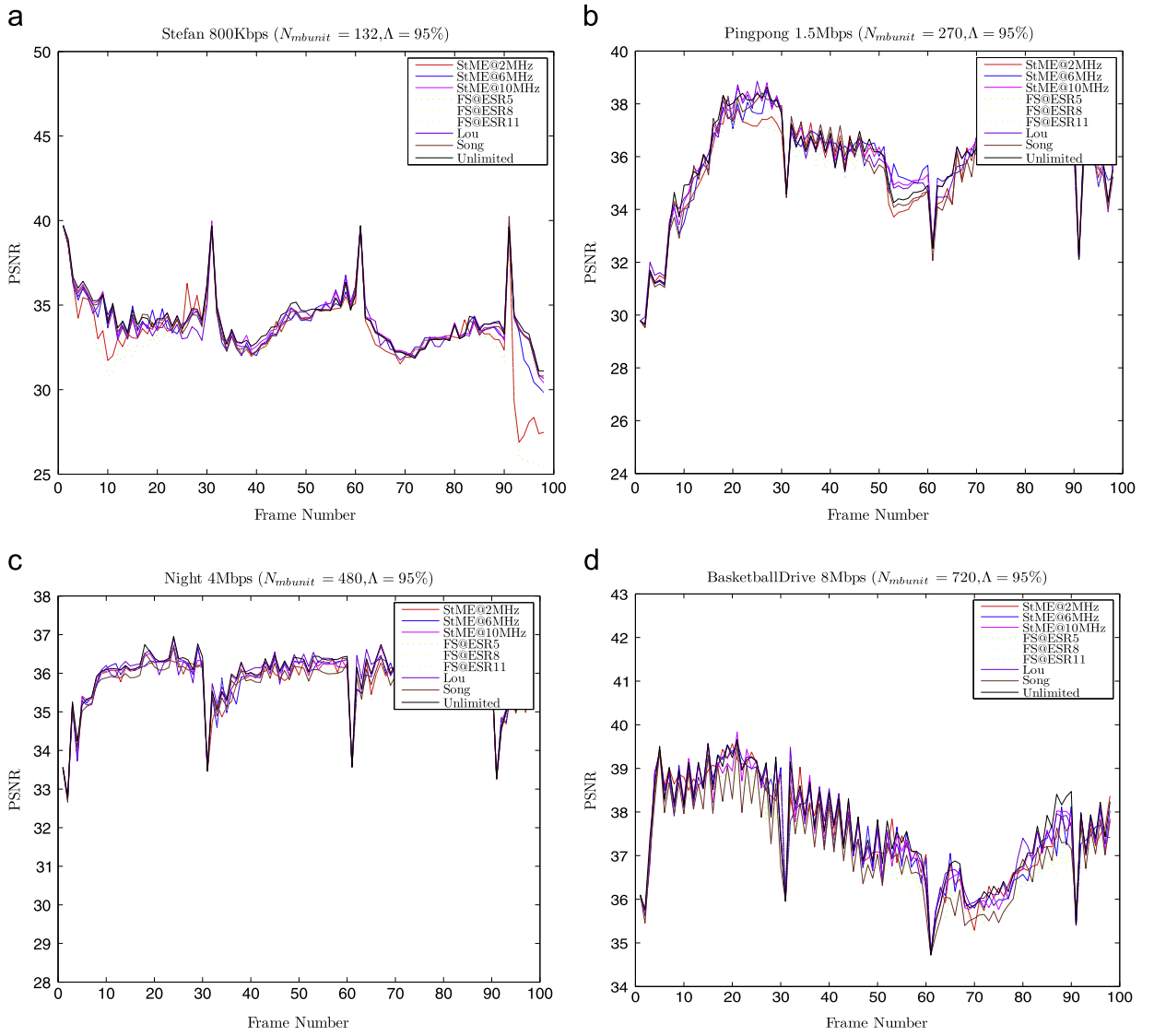

Fig. 9. PSNR curves for different DSR algorithms. (a) *Stefan* 800Kbps with $N_{mbunit} = 132$ and $\Lambda_{TH} = 95\%$. (b) *Pingpong* 1.5 Mbps with $N_{mbunit} = 270$ and $\Lambda_{TH} = 95\%$. (c) *Night* 4 Mbps with $N_{mbunit} = 480$ and $\Lambda_{TH} = 95\%$. (d) *BasketballDrive* 8 Mbps with $N_{mbunit} = 720$ and $\Lambda_{TH} = 95\%$.

table, we can see that the proposed CDSR algorithm achieved about 0.1–0.3 dB average PSNR improvement when the computation consumption is restricted to a specific level as compared with its equivalent Fixed SR algorithm. The higher the motion of the sequence is, the larger the video quality improvement represented by the ΔESR can be achieved, for example, an increase of 0.27 dB was observed for the *BasketballDrive* sequence. The quality degradation compared with the computation-unlimited DSR algorithm ΔUN decreases with the increase in the hardware performance. For the ME with high performance PE engine (e.g., HiME@100Mhz for 720P), only 0.04 dB quality degradation on average is brought by the proposed CDSR algorithm which is negligible compared with the computation-unlimited DSR algorithms. The average SR for the CDSR algorithm is below its equivalent Fixed SR. The WCRs of the proposed CDSR algorithm are always less than 100% and they are much smaller than the unlimited and Lou's algorithm. Although Song's algorithm shows low ASRs and WCRs for the 720P and 1080P sequences, the computation reduction is achieved at the expense of RD performance degradation compared with other DSRs. With the WCR values of the proposed CDSR and the benchmarks, the computation reduction efficiency can be evaluated by the following equations:

$$\frac{(\mathrm{WCR}_{\mathrm{benchmark}} - \mathrm{WCR}_{\mathrm{CDSR}})}{\mathrm{WCR}_{\mathrm{benchmark}}}. \tag{30}$$

It can achieve about 50–90% computation savings compared with the benchmarks, and the video quality of the proposed algorithm improves with the increase of

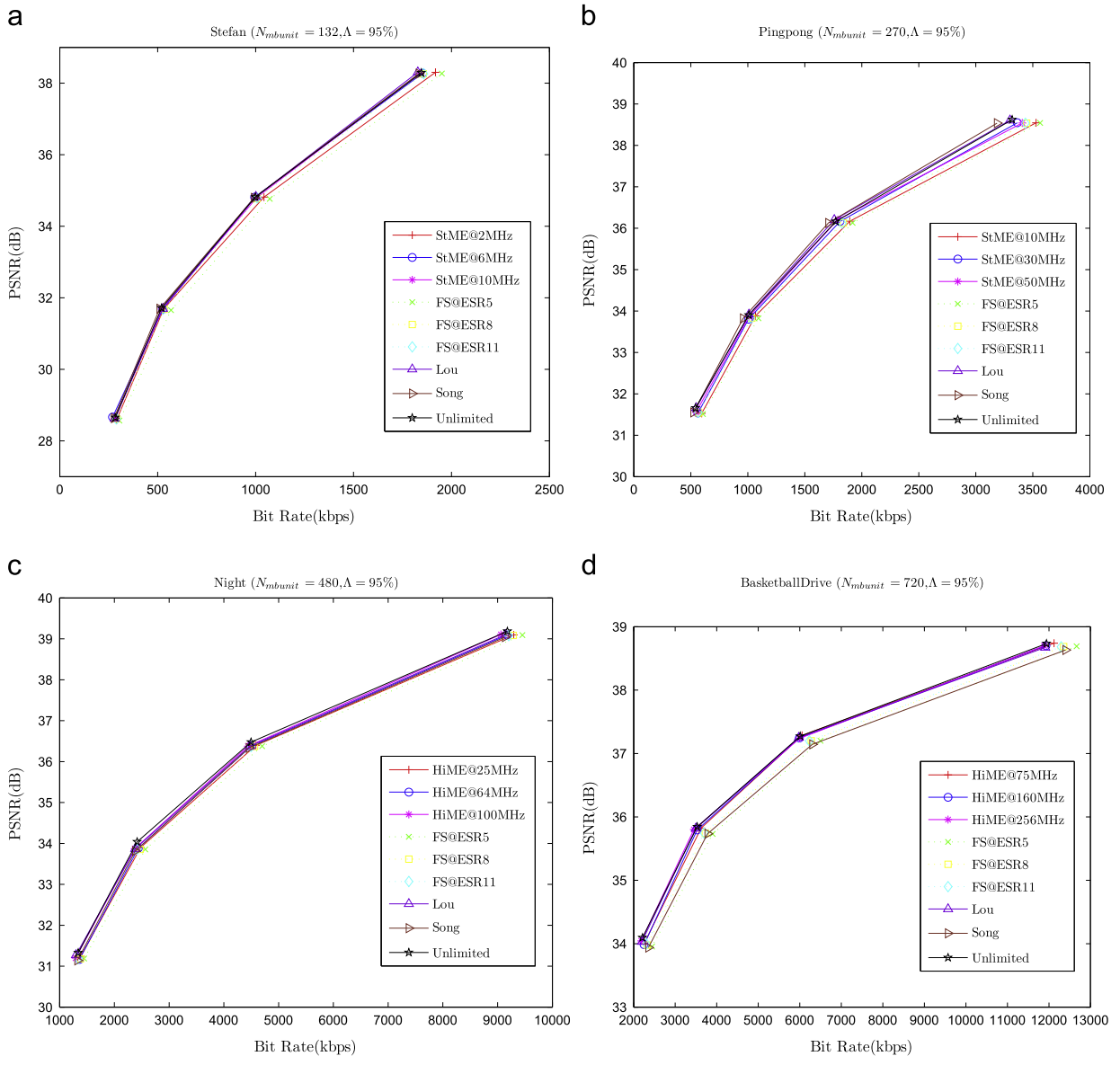

Fig. 10. *R–D* curves for different DSR algorithms. (a) *Stefan* with $N_{mbunit} = 132$ and $\Lambda_{TH} = 95\%$. (b) *Pingpong* with $N_{mbunit} = 270$ and $\Lambda_{TH} = 95\%$. (c) *Night* with $N_{mbunit} = 480$ and $\Lambda_{TH} = 95\%$. (d) *BasketballDrive* with $N_{mbunit} = 720$ and $\Lambda_{TH} = 95\%$.

hardware resources. As a result, it is verified that the proposed CDSR algorithm is an effective method to manage the computation consumption of the DSR algorithm while keeping similar RD performance.

5. Conclusion

DSR is a commonly used method to reduce computational complexity of ME. In this paper, we presented an effective PMV deviation metric to model the relationship between SR and MVD distribution according to the prediction differences of both temporal and spatial neighboring motions. In addition, a computation-constrained DSR control algorithm is proposed to manage the computational complexity while maximizing video coding quality in a real-time computational constrained scenario. The SR is dynamically determined by three factors: motion complexity, user-defined probability and computation budget. The proposed CDSR is an effective and quantifiable algorithm to allocate more computation budget to the blocks with high PMV deviations, and less computation budget to the well-matched motion predicted blocks, while maintaining a constrained computation requirement. According to the experimental results, the proposed CDSR control algorithm is proven to be an effective method to manage the computation consumption of the DSR algorithm while keeping similar RD performance. It can achieve about 0.1–0.3 dB average PSNR improvement when the computation consumption is restricted to a specific level as compared with its equivalent

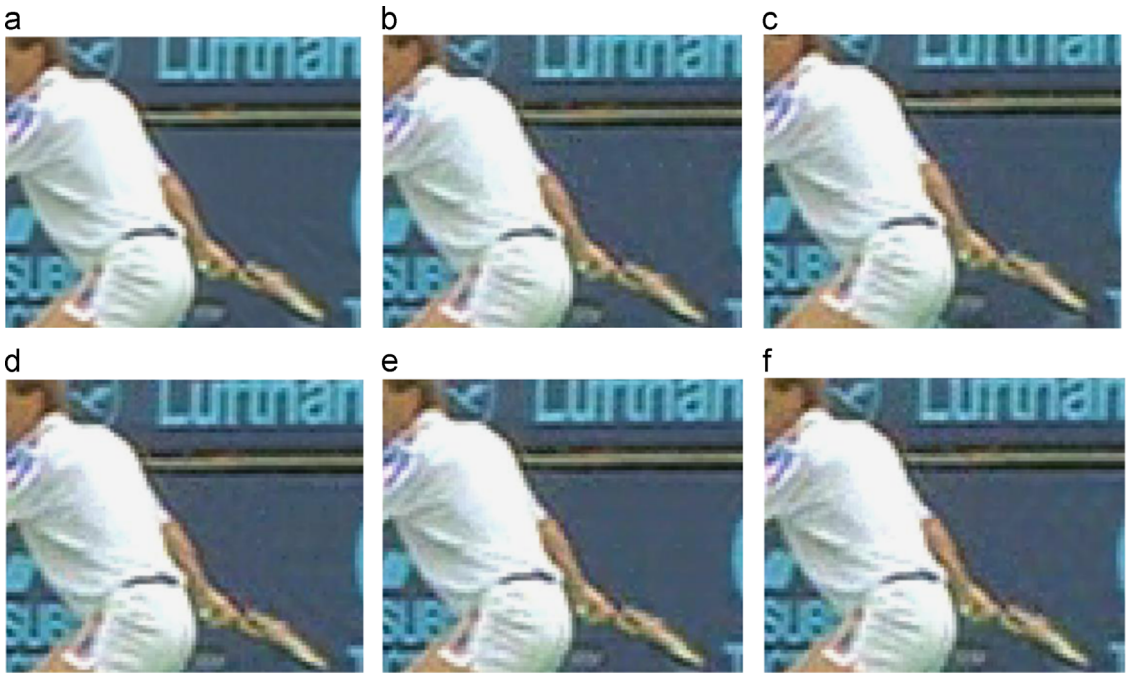

Fig. 11. Visual comparison of the benchmarks and the proposed CDSR. (a) Ground truth. (b) FS@ESR8. (c) the result of Lou. (d) the result of Song. (e) the result of the proposed CDSR with unlimited computing resources. (f) the result of the proposed CDSR with StME@6Mhz.

Table 6
Experimental results for different DSR algorithms.

Sequence	ME engine	Equivalent fix SR	CDSR				Unlimited			Lou			Song		
			ΔESR (dB)	ΔUN (dB)	ASR	WCR (%)	ΔESR (dB)	ASR	WCR (%)	ΔESR (dB)	ASR	WCR (%)	ΔESR (dB)	ASR[a]	WCR (%)
Stefan (CIF)	StME@2Mhz	[−5, 4]	+0.27	−0.17	4.5	92	+0.44	8.6	918	+0.42	6.1	1242	+0.49	14.2	807
	StME@6Mhz	[−8, 7]	+0.10	−0.03	6.2	90	+0.13		358	+0.11		485	+0.17		315
	StME@10Mhz	[−11, 10]	+0.07	−0.02	7.3	82	+0.09		189	+0.07		257	+0.14		167
Pingpong (SD)	StME@10Mhz	[−5, 4]	+0.11	−0.28	4.2	96	+0.39	9.8	2360	+0.42	8.6	2775	+0.49	11.9	566
	StME@30Mhz	[−8, 7]	+0.09	−0.11	5.7	99	+0.20		922	+0.23		1084	+0.30		221
	StME@50Mhz	[−11, 10]	+0.12	−0.05	7.1	99	+0.17		488	+0.21		573	+0.28		117
Night (720P)	HiME@25Mhz	[−11, 10]	+0.15	−0.19	9.8	94	+0.34	23.8	472	+0.27	26.3	571	+0.21	10.4	89
	HiME@64Mhz	[−16, 15]	+0.07	−0.14	13.8	86	+0.21		223	+0.14		270	+0.08		42
	HiME@100Mhz	[−22, 21]	+0.09	−0.04	18.0	81	+0.14		118	+0.07		143	+0.01		22
BasketballDrive (1080P)	HiME@75Mhz	[−11, 10]	+0.27	−0.05	9.1	82	+0.32	26.0	950	+0.30	24.9	815	+0.05	9.9	81
	HiME@160Mhz	[−16, 15]	+0.16	−0.04	15.3	99	+0.20		449	+0.19		385	−0.06		38
	HiME@256Mhz	[−22, 21]	+0.17	−0.01	19.7	98	+0.18		237	+0.17		204	−0.09		20

[a] 3-step search range modification stage is included for the ASR calculation.

Fixed SR algorithm and can achieve about 50–90% computation savings when compared to the benchmarks.

Acknowledgements

This work is partially supported by grants from the Chinese National Natural Science Foundation under contract no. 61171139 and no. 61035001, and National High Technology Research and Development Program of China (863 Program) under contract no. 2012AA011703.

References

[1] Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264|ISO/IEC 14496-10 AVC), ITU-T|ISO/IEC, 2003.
[2] T. Wiegand, G.J. Sullivan, G. Bjøntegaard, A. Luthra, Overview of the H.264/AVC video coding standard, IEEE Trans. Circuits Syst. Video Technol. 13 (7) (2003) 560–576. (Jul.).
[3] T. Sikora, MPEG digital video-coding standards, IEEE. Signal Process. Mag. 14 (5) (1997) 82–100. (Sep.).
[4] I. Ismaeil, A. Docef, F. Kossentini, R. Ward, Efficient motion estimation using spatial and temporal motion vector prediction, Proc. Int. Conf. Image Process 1 (1999) 70–74.

[5] J. Liu, B. Feng, Z. Ma, W. Liu, Adaptive motion vector pediction based on spatiotemporal correlation, in: Proc. IEEE International Conference on Wireless Communications, Networking and Mobile Computing, 2006, 1–4.
[6] T.C. Chen, S.Y. Chien, Y.W. Huang, C.H. Tsai, C.Y. Chen, T.W. Chen, L.G. Chen, Analysis and architecture design of an HDTV720p 30 frames/s H.264/AVC encoder, IEEE Trans. Circuits Syst. Video Technol. 16 (2006) 673–688.
[7] Y.W. Huang, C.Y. Chen, C.H. Tsai, C.F. Shen, L.G. Chen, Survey on block matching motion estimation algorithms and architectures with new results, J. VLSI Sig. Process. Syst. Sig. Image Video Technol. 42 (2006) 297–320.
[8] T. Koga, K. linuma, A. Hirano, Y. Iijima, T. Ishiguro, Motion compensated inter frame coding for video conferencing, in: Proc. Nat. Telecommun. Conf., 1981, C9.6.1–C9.6.5.
[9] L.M. Po, W.C. Ma, A novel four-step search algorithm for fast block motion estimation, IEEE Trans. Circuits Syst. Video Technol. 6 (3) (1996) 313–317.
[10] J.Y. Tham, S. Ranganath, M. Ranganath, A.A. Kassim, A novel unrestricted center-biased diamond search algorithm for block motion estimation, IEEE Trans. Circuits Syst. Video Technol. 8 (4) (1998) 369–377.
[11] C. Zhu, X. Lin, L.P. Chau, Hexagon-based search pattern for fast block motion estimation, IEEE Trans. Circuits Syst. Video Technol. 12 (2002) 349–355. (May).
[12] M.C. Hong, H.H. Oh, Range Decision for Motion Estimation of VCEG-N33, JVT B022, Switherland, February 2002.
[13] X. Xu, Y. HE, Modification of Dynamic Search Range for JVT, JVT Q088-L, USA, Oct. 2005.
[14] T. Yamada, M. Ikekawa, I. Kuroda, Fast and accurate motion estimation algorithm by adaptive search range and shape selection, Proc. ICASSP 2 (2005) 897–900. (March).
[15] L. Zhang, W. Gao, Improved FFSBM algorithm and its VLSI architecture for variable block size motion estimation of H.264, Proc. IEEE Int. Symp. Intell. Signal Process. Commun. Syst. (2005) 445–448. (Dec.).
[16] T. Shimizu, A. Yoneyama, Y. Takishima, Dual-path block size decision for fast motion search in H.264/AVC, Proc. SPIE Conf. Visual Commun. Image Process. 5960 (2005) 48–56. (June).
[17] T. Song, K. Ogata, K. Saito, T. Shimamoto, Adaptive search range motion estimation algorithm for H.264/AVC, Proc. IEEE ISCAS (2007) 3956–3959.
[18] Y.H. Ko, H.S. Kang, S.W. Lee, Adaptive search range motion estimation using neighboring motion vector differences, IEEE Trans. Consum. Electron. 57 (2) (2011) 726–730. (May).
[19] W.e.i. Dai, Oscar C. Au, Sijin Li, L.i.n. Sun, Ruobing Zou, Adaptive search range algorithm based on cauchy distribution, IEEE Int. Conf. Visual Commun. Image Process. (2012) 1–5.
[20] C. Lou, S. Lee, C.J. Kuo, Motion vector search window prediction in memory-constrained systems, in: Proc. XXXIIth SPIE Applicat. Digital Image Process., 7443, 74430E, 2009.
[21] Chung-Cheng Lou, Szu-Wei Lee, C.-C. Jay Kuo, Adaptive motion search range prediction for video encoding, IEEE Trans. Circuits Syst. Video Technol. 20 (2010) 1903–1908. (Dec.).
[22] S.Y. Yap, V. McCanny, A VLSI architecture for variable block size video motion estimation, IEEE Trans. Circuits Syst. Video Technol. 51 (7) (2004) 384–389.
[23] C.Y. Chen, S.Y. Chine, Y.W. Hung, T.C. Chen, T.C. Wang, L.G. Chen, Analysis and architecture design of variable block-size motion estimation for H.264/AVC, IEEE Trans. Circuits Syst. Video Technol. 53 (2) (2006) 578–593.
[24] H.B. Yin, H.Z. Jia, H.G. Qi, X.H. Ji, X.D. Xie, W. Gao, A hardware-efficient multi-resolution block matching algorithm and its VLSI architecture for high definition MPEG-like video encoders, IEEE Trans. Circuits Syst. Video Technol. 20 (9) (2010) 1242–1254. (Jul.).
[25] S. Mochizuki, T. Shibayama, M. Hase, F. Izuhara, K. Akie, M. Nobori, R. Imaoka, H. Ueda, K. Ishikawa, H. Watanabe, A 64 mW high picture quality H.264/MPEG-4 video codec IP for HD mobile applications in 90 nm CMOS, IEEE J. Solid-State Circuits 43 (2008) 2354–2362.
[26] L.-F. Ding, W.-Y. Chen, P.-K. Tsung, T.-D. Chuang, P.-H. Hsiao, Y.-H. Chen, H.-K. Chiu, S.-Y. Chien, L.-G. Chen, A 212 MPixels/s 4096 × 2160p multiview video encoder chip for 3D/Quad full HDTV applications, IEEE J. Solid-State Circuits 45 (1) (2010) 46–58.
[27] J.-J. Tsai, H.-M. Hang, Modeling of pattern-based block motion estimation and its application, IEEE Trans. Circuits Syst. Video Technol. 19 (2009) 108–113.
[28] L. Yilong, S. Oraintara, Complexity comparison of fast block-matching motion estimation algorithms, IEEE Int. Conf. Acoust. Speech Signal Process. 3 (2004) 341–344.
[29] N. Kamaci, Y. Altunbasak, R.M. Mersereau, Frame bit allocation for the H.264/AVC video coder via cauchy-density-based rate and distortion models, IEEE Trans. Circuits Syst. Video Technol. 15 (2005) 994–1006.
[30] J. Peacock, Two-dimensional goodness-of-fit testing in astronomy, Mon. Not. R. Astron. Soc. 202 (1983) 615–627.
[31] ⟨http://www.broadcom.com/products/Cable/MPEG-2-Digital-Audio-Video-Encoders/BCM7042⟩.
[32] ⟨http://www.imaginecommunications.com/pdfs/datasheets/IBS_DataSheet_v4.0_1010.pdf⟩.
[33] K.B. Lee, T.C. Lin, C.W. Jen, An efficient quality-aware memory controller for multimedia platform SoC, IEEE Trans. Circuits Syst. Video Technol. 15 (5) (2005) 620–633.
[34] P.-K. Tsung, W.-Y. Chen, L.-F. Ding, S.-Y. Chien, L.-G. Chen, Cache-based integer motion/disparity estimation for quad-HD H.264/AVC and HD multiview video coding, Proc. ICASSP (2009) 2013–2016.
[35] J.-C. Tuan, T.-S. Chang, C.-W. Jen, On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture, IEEE Trans. Circuits Syst. Video Technol. 12 (1) (2002) 61–72.
[36] C.Y. Chen, C.T. Huang, Y.H. Chen, L.G. Chen, Level C+data reuse scheme for motion estimation with corresponding coding orders, IEEE Trans. Circuits Syst. Video Technol. 16 (2006) 225–558.
[37] N. Roma, L. Sousa, Efficient and configurable full search block matching processors, IEEE Trans. Circuits Syst. Video Technol. 12 (2002) 1160–1167.
[38] G. Bjontegaard, Calculation of average PSNR difference between RDcurves, in: Proc. ITU-T Q.6/SG16 VCEG 13th Meeting, Austin, TX, Apr. 2001, document VCEG-M33.