Complexity-Constrained H.264 Video Encoding

Li Su, Yan Lu, *Member, IEEE*, Feng Wu, *Senior Member, IEEE*, Shipeng Li, *Member, IEEE*, and Wen Gao, *Senior Member, IEEE*

***Abstract*—In this paper, a joint complexity-distortion optimization approach is proposed for the real-time H.264 video encoding under the power-constrained environment. The power consumption is first translated to the encoding computation costs measured by the number of scaled computation unit (CU) consumed by basic operations. The solved problem is then specified to be the allocation and utilization of the computational resources. A computation allocation model (CAM) with virtual computation buffers is proposed to optimally allocate the computational resources to each video frame. In particular, the proposed CAM and the traditional hypothetical reference decoder (HRD) model have the same temporal phase in operations. Further, to fully utilize the allocated computational resources, complexity configurable motion estimation (CAME) and complexity configurable mode decision (CAMD) algorithms are proposed for H.264 video encoding. In particular, the CAME is performed to select the path of motion search at the frame level, and the CAMD is performed to select the order of mode search at the macroblock level. Based on the hierarchical adjusting approach, the adaptive allocation of computational resources and the fine scalability of complexity control can be achieved.**

***Index Terms*—Real-time video coding, power consumption, complexity control, H.264/AVC.**

I. Introduction

WITH rapid increases in portable devices and network bandwidth, real-time visual communications over wireless networks are generally not restricted by bandwidth availability. However, computational costs (i.e., processor workloads) to encode the video content become the bottleneck, mainly due to the constraint of power supply. Typically, encoding and transmission cause most of the power consumption of portable devices in wireless visual communications [1]. The former is monotonic ascending with encoding complexity, and the latter is monotonic ascending with the compressed bit-rate [2], [3]. In the practical applications under bandwidth constraint, the target bit-rate might be preset and adaptively satisfied by some efficient rate control algorithm [4]. In this case, the power consumption due to transmission is stable, while the power consumption due to encoding can be adaptively adjusted. Therefore, the power consumption can be translated to the computational costs.

Computational costs (i.e., processor workloads) to encode frames of a video content typically vary. This is potentially problematic. When necessary computational resources are not available for real-time video encoding, it generally causes computation overflow, frame dropping, and the introduction of jitter (transmission delays) into a video stream, resulting in low-quality video playback. In other words, there exists tradeoff between the coding complexity and the compression efficiency judged by rate-distortion (R-D) performance. Referring to the multiple objective optimization model proposed by Ahmad *et al.* [5], this optimization problem can also be formulated as:

$$\min \begin{pmatrix} C_e(R,S) \\ D_e(R,S) \end{pmatrix}, \quad \text{s.t.} \quad C_e(R,S) \le C_{\text{cst}}, \qquad (1)$$

where $C_e(R,S)$ and $D_e(R,S)$ denote the computational consumption and video distortion with encoding parameter set S at rate R, and C_{cst} is the available computation capability constraint. For this multiple objective optimization problem, there is no unique solution [6]. Hence, the problem becomes how to achieve the trade-off between the distortion and the computational consumption. In general, we have to answer two questions here. The first one is how to allocate the overall available computational resources to different frames and/or coding modules. In particular, at any time the workload of processor should not exceed a given upper bound. The second one is how to efficiently utilize the allocated computational resources by adjusting encoding parameters. In particular, the sacrifice of coding efficiency due to the limitation of computational resources should be as small as possible. Currently, many optimization techniques are developed to enhance the video encoding, including algorithmic optimization, compiler optimization and code optimization. For example, Akramullah *et al.* explore the performance tradeoffs and benchmarking by using various techniques separately or jointly [7]. In this paper, we only focus on the algorithmic optimization, which could achieve quantitative adjusting and is more independent of specific source codes.

Most related work is based on the establishment of some mathematic model so as to accurately describe the power-rate-distortion (P-R-D) relationship. For example, Eisenberg *et al.* minimize the transmission energy under a given end-to-end distortion constraint by jointly considering the packet loss in channel and the error concealment in decoder [8]. However, with the increasing of the number of optional encoding modes in one macroblock (MB) or packets in one frame, the number of the proposed source coding tree branches

Manuscript received March 28, 2007; revised November 26, 2007. .

L. Su is with the Graduate School of the Chinese Academy of Sciences, Beijing 100080, China (e-mail: lsu@jdl.ac.cn).

Y. Lu is with Microsoft Research Asia, Beijing 100080, China (e-mail: yanlu@microsoft.com).

F. Wu is with Microsoft Research Asia, Beijing 100080, China (e-mail: fengwu@microsoft.com).

S. Li is with Microsoft Research Asia, Beijing 100080, China (e-mail: spli@microsoft.com).

W. Gao is with the School of Electronic Engineering and Computer Science, Peking University, Beijing 100080, China (e-mail: wgao@jdl.ac.cn).

This work was performed at Microsoft Research Asia, Beijing, China, and was supported in part by National Natural Science Foundation of China under Grand 60736043. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Ishfaq Ahmad.

Digital Object Identifier 10.1109/TCSVT.2009.2014017

increases with geometric series as well. He *et al.* develop a P-R-D analysis framework based on the MPEG-4 video encoder [2], which has extended the traditional R-D analysis by including the power consumption. The P-R-D performance is jointly optimized by adjusting some complexity control parameters such as the number of sum of absolute differences (SAD) computations, the number of DCT computations and frame rate. In order to accurately describe the P-R-D relationship by a universal model, a number of adjustable parameters are necessary due to the diversity of video contents, which lead to significant overhead of computational costs. To reduce the overhead, the recommended frequency of parameter adjustment is usually low, e.g., once per 5 seconds in [2]. In summary, while the parametric model provides a good solution for the above two problems, it also presents inherently high overhead of computing in setting up a general and accurate P-R-D model.

Moreover, some non-parametric power control methods with high frequency of parameter adjustment and low overhead of computational costs have been proposed. For example, Agrawal *et al.* propose to reduce the energy consumption of visual communications by reducing the average bit-rate and discarding some selected packets [9]. However, the bit-rate reduction and frame dropping result in global deterioration of video quality. Li *et al.* classify the video content into foreground and background regions based on motion activity detection. More bits and more computational resources are allocated to the foreground to enhance its video quality [10]. Considering the high overhead of computational costs, the region classification may not be very accurate. Thus the adjustability range of power consumption, which relies on the accuracy of region classification, is then limited. In summary, these non-parametric methods own the advantages of low overhead of parameter adjustment in complexity control. However, they usually have low overall R-D performances.

More recently, the complexity control issue has been studied in terms of the up-to-date video coding standard H.264/AVC. Some new coding tools in H.264 greatly improve the coding efficiency but result in the complexity increase dramatically. When the previous complexity control algorithms are extended to the H.264 video encoding, it is hard to utilize the advantages of these new coding tools. According to the statistics in the literature, the R-D optimized coding mode decision and the variable block-size motion estimation (ME) are the two main time-consuming modules in a typical H.264 encoder [11]–[13]. Many fast algorithms have been proposed for the complexity reduction of motion estimation and/or coding mode decision [14]–[18]. The power constraint is jointly considered in some fast algorithms. For example, Liang *et al.* propose the optimization strategy to start the coding from the highest complexity level and automatically migrates to a lower or higher level by greedy motion estimation control [5], [17] and zero quantized macroblocks early detection [5] or frame rate adjusting [17], until power constraint and performance improvement get satisfied. However, these greedy algorithms are hard to achieve arbitrary computation constraint directly. Kannangara *et al.* develop a complexity reduction algorithm by identifying the MBs that are likely to be skipped prior to motion estimation [18]. By adjusting the *SKIP* mode proportion, it can achieve arbitrary computation constraint. However it would hurt the visual quality and influence the rate control.

In view of the above problems, we propose an efficient joint complexity-distortion optimization video coding scheme under the power constraint. In order to reduce the overhead of power control, power constraints are translated to the encoding computation costs, which are measured by the number of scaled SAD operations. Firstly, for the computational resources allocation, we propose a computation allocation model (CAM) with virtual computation buffers (VCB) to facilitate the optimal allocation of restricted computational resources to each frame. Inspired by the leaky bucket model in hypothetical reference decoder (HRD), we also employ a virtual leaky bucket model here. Actually, the scheduling of computation costs for each frame with the proposed CAM has the identical temporal phase to the scheduling of coded bits for each frame with the HRD.

Further, for the allocated computational resources fully utilization, we propose a complexity-configurable H.264 video encoding scheme with complexity control on motion estimation and mode decision. The complexity-adjustable motion estimation (CAME) algorithm is performed at the frame-level to decide the complexity level for ME. In particular, the selection of the motion estimation search path and the termination point is based on the rate-distortion cost function and the allocated computation budget. Then, the complexity-adjustable mode decision (CAMD) algorithm is performed at the block-level to select the search order and the termination point of the R-D optimized coding mode decision. In particular, the complexity level decided in the CAMD is based on the simple spatial and temporal correlation statistics and the available computation budget. Based on the hierarchical adjusting scheme, the adaptive allocation of computational resources and the fine scalability of complexity control are achieved.

The rest of this paper is organized as follows. In Section II, we analyze the relation between power consumption and encoding complexity in terms of a typical H.264 encoder. In Section III, we describe the proposed CAM with VCB for computation resource scheduling and allocation. In Section IV, we propose a video coding scheme with scalable computational complexity control on motion estimation and mode decision. The experimental results are given in Section V. Finally, Section VI concludes this paper.

II. H.264 Encoding Complexity Analysis

A. Modules Analysis

It has been commonly recognized that the most computation-consuming modules in a typical H.264 video encoder include the ME with fractional motion vector (MV) precision and the R-D optimized coding mode decision with variable block-size ME [11]–[16]. For example, ME with quarter-pixel precision typically consumes 60% (with 1 reference frame) to 80% (with 5 reference frames) of the total encoding time [14], and the percentage becomes even larger when the search range increases. The benefit is that the fractional-pixel ME can reduce the bit-rates up to 30% except at the very low bit-rates, while it increases the processing time about 10% compared with the integrate-pixel ME [12]. Moreover, the complexity increases in direct proportion to the number of modes used for the variable block-size ME, while the R-D performance gain mainly comes from some typical modes. For example, about 60%–85% probability of PSNR improvement and 75%–85% of the bit-rate reduction can already be achieved using modes P16 $\times$ 16 to P8 $\times$ 8 [12].

In summary, the diversity of the operation configuration for motion estimation (e.g., the MV precision) and mode decision (e.g., the number of candidate modes) have great effects on the variety of encoding complexity. In order to achieve more flexible complexity control, we decompose these modules into a number of operational sets, and then introduce the complexity level (CL) to represent the different set of candidate encoding operations under a restriction of available computation resource. For example, it represents different searching path and stop point for motion estimation module, as shown in Table III. For mode decision module, it represents different candidate modes and searching order, as shown in Table V. It should be noted that different frames with the same complexity level configuration would consume different computational resources due to the diversity of video contents. In order to achieve most efficient combination of each encoding module under a give complexity constraint, a universal and quantitative measurement of the computational consumption for arbitrary complexity level of different modules is the coming problem.

B. Quantitative Measurement

As described in the literature [2], [19], [20], using dynamic voltage scaling (DVS) technology in CMOS circuit design, the power consumption of computational operations is a monotonic ascending function of the number of processor cycles per second. Accordingly, for a given power supply constraint of a certain portable device, the encoding complexity constraint becomes a function of the number of processor cycles [2], [3]. To the benefit of the generality as well as the convenience of further application on power-aware video coding, the computational operations of a typical H.264 encoder are quantitatively analyzed in terms of the number of processor cycles consumed by basic operation units.

In this paper, we employ the CU to facilitate the quantitative measurement of the real-time computational consumptions. Firstly, we divide the high-level encoding module into a number of basic operation units. Each operation unit contains a fixed number of processor circles. For example, the motion estimation module can be taken as the combination of a number of SAD operations. We measure the encoding complexity of a basic operation unit by summing up all the required processor cycles for its sub-functions, based on the detailed analysis of operations of addition, multiplication, shift, etc., [3]. Further, we define the CU to be the computational consumption by the SAD operated on a 4 $\times$ 4 block, and then scale the computation consumptions of other basic operation units as the multiple of CU, as shown in Table I. According to Table I, we can easily achieve the required number of CUs at a particular complexity level for the real-time video encoding.

We choose the computational consumption of 4 $\times$ 4 SAD as the CU because it is the most frequently performed sub-function with specific physical meaning during encoding. The benefit

TABLE I
SUMMARY OF COMPUTATIONAL CONSUMPTION FOR BASIC OPERATION UNITS

Sub-function		Cycles	CU
SAD_4x4		353	1.0
DCT_4x4		879	2.5
Luma interpolation_16x16		41961	118.9
Chroma interpolation_16x16		58337	165.3
MV initialization		356	1.0
Motion compensation		36	0.1
Fast integer-pixel ME_16x16		22537	63.8
Fast sub-pixel ME_16x16		3650	10.3
Inter prediction	SKIP_16x16	1284	3.6
	P_16 x 16	6932	19.6
	P_16 x 8	3644	10.3
	P_8 x 16	3644	10.3
	P_8 x 8	2000	5.7
	P_8 x 4	1178	3.4
	P_4 x 8	1178	3.4
	P_4 x 4	767	2.2
Intra prediction	I_16 x 16	107856	288.0
	I_4 x 4	12088	34.2
De-blocking filter_16x16		2798	8.0

*Note: The postfix "_MxN" indicates the count unit based on integer-pixel.

is to decrease the count frequency and value to trace real-time computational costs of actual processor cycles. Moreover, since the quantitative analysis in theory is based on the basic operation unit and their performing frequencies, it would be more flexible and universal. For example, in practical applications, if any other optimization technology such as compiler optimization is employed to decrease the required number of processor cycles consumed by some basic operations, the quantitative measurement of practical computational costs only needs to update the number of CUs corresponding to those basic operations in Table I.

III. Computation Allocation Model

In this section, we propose a computation allocation model with virtual computation buffers. It is used for the optimal allocation of the available computational resources to each frame. For the convenience of description, some abbreviations and symbols are employed as shown in Table II.

A. Problem Statement

As for the allocation of computational resources, the key point is how to avoid the overflow or underflow of the computation. The overflow occurs as follows. If the actual encoding of a frame costs too many computational resources, the encoding delay may exceed the given maximum delay of transmission/rendering. In this case, jitters occur due to the frame dropping. The underflow occurs as follows. If the actual encoding of a frame costs too few computational resources, the processor may be in the idle status for some time before the next frame arrives. In the practical application, making the processor idle could save more power consumption. However, in theory, more computation resources could result in better R-D performance in general. In terms of the optimization of the overall coding efficiency, as well as to the benefit of generality in presenting our model, we choose to avoid the idleness of encoder for the possibility of better performance in the case of

TABLE II
SUMMARY FOR ABBREVIATIONS AND SYMBOLS

Abbreviation / Symbols	Explanation	Abbreviation / Symbols	Explanation
HRD	hypothetical reference decoder	CAM	computation allocation model
PCB	coded picture buffer	VCB	virtual computation buffers
CBR	constant bit-rate	CCR	constant computation rate
VBR	variable bit-rate	VCR	variable computation rate
BOB	virtual encoder bit-stream output buffer	BIB	virtual decoder bit-stream input buffer
Fr	the frame rate	Cr	the encoder computation rate
δ	the transmission delay time for coded bit-stream	R	the transmission bit-rate
D_e, D_d, D	the maximum encoding / decoding delay time	B_e, B_d, B_M	the maximum buffer size of BOB/ BIB
S_n	the inputted data of the n^{th} original frame	b_n	the number of bits for the coded n^{th} frame
C_n	the buffer fullness of VCB when the n^{th} original frame arrives	c_n	the required computation of the originally input n^{th} frame
C_M	the maximum buffer size of VCB	C_h, C_l	the upper/lower bounds line of CAM
$C_{n,\max}, C_{n,\min}$	the required maximum / minimum computational costs for coding the n^{th} frame	U_n, L_n	the upper/lower bound of the CAM for the n^{th} frame
$CL_{n,est}$	the estimation of the complexity level for the n^{th} frame	$T_a(n)$	the time when the n^{th} frame arrives at the VCB of the encoder
$T_s(n)$	the time to start the encoding of the n^{th} frame	$T_e(n)$	the time to end the encoding of the n^{th} frame
$T_r(n)$	the time to remove the coded n^{th} frame from the BOB	$T_r'(n)$	the time to remove the coded n^{th} frame from the BIB
$c^i_{n,used}$	the actual computational consumption for coding the i^{th} MBs in the n^{th} frame	$c_{n,used}$	the actual computational consumption for coding the whole n^{th} frame
$c^i_{n,alloc}$	the allocated computation for the i^{th} MB in the n^{th} frame	$c_{n,alloc}$	the allocated computation for the whole n^{th} frame

sufficient computational resources (In practice, we could certainly let the encoder idle for more power saving by adjusting the lower bound of the proposed CAM).

B. Virtual Buffers

In particular, we employ a virtual computational resources buffer at the encoder side to describe the scheduling and allocation of the computational resources. Inspired by the leaky bucket model of HRD, we also employ a virtual leaky bucket model to describe the temporally changed condition in terms of the computational consumption. Moreover, we define two virtual bits buffers, i.e., the encoder bit-stream output buffer (BOB) and the decoder bit-stream input buffer (BIB), to describe the coded bits scheduling of the encoder output and the decoder input, respectively. The relationships between HRD and CAM virtual buffers as well as operations on these buffers are described in detail as below.

Fig. 1(a) shows a generalized HRD leaky bucket model in the case of constant bit-rate (CBR) coding. Note that, the decoder bits buffer fullness (the shaded area in the right part) is the complement of the encoder bits buffer fullness (the shaded area in the left part). More details could be found in [21].

Fig. 1(b) shows the proposed BOB, which is assumed at the output side of the encoder. Assume that the original frames of a video is fed into the encoder with a constant frame rate Fr. In particular, for the inputted data S_n of the nth original frame, its coded frame with the number of bits b_n is moved out of the encoder after a constant delay D_e. In order to avoid the underflow and overflow of the first coded frame, the maximum coding delay D_e must satisfy $b_0/R \leq D_e \leq B_e/R$, where B_e is the maximum buffer size of BOB and R is the average output bit-rate.

Fig. 1(c) shows the proposed BIB, which is assumed at the input side of decoder. Assume that the movement of the coded bits b_n for the whole nth frame from BIB to decoder is instantly completed at the moment $T_r'(n)$. It is obvious that BIB is identified with CPB in HRD. Note that, the complement of the BOB fullness is just a horizontal mirror of the BIB fullness. Assume D_d is the maximum decoding delay time. In order to avoid the underflow and overflow of the first coded frame, D_d must satisfy that: $b_0/R \leq D_d \leq B_d/R$, where B_d is the max buffer size of BIB and R is the average input bit-rate. For the purpose of simplification, we could assume that: $B_e = B_d = B_M$, and $D_e = D_d = D$.

Fig. 1(d) shows the leaky bucket bounds of the proposed CAM in the case of constant computation rate. Assume the computational cost c_n consumed by the coding of this frame is added to the total computational cost instantly. Line C_l is the lower bounds, and the state below this bound indicates the VCB is idle (underflow). Line C_h is the upper bounds, and the state above this bound indicates the VCB is overflow. The slope indicates the computation rate Cr. It should be noted that here Cr is the peak computation rate, and it is constant for a settled hardware platform. Though the total amount of power supply is decreasing with the lapse of time, but the computation property is relative stable during a period. Therefore, we can describe the model under the assumption of constant computation rate. As for the case of variable computation rate, we can regard it as piecewise constant computation rate.

Fig. 1(e) shows the proposed virtual computational resource buffer at the encoder side. Assume that the maximum buffer size of VCB is C_M, the average encoder computation rate is Cr, and the maximum encoding delay is D. Then, $C_M = D \cdot C_r$. Further assume that: $T_a(n)$ is the time when the nth frame arrives at the VCB of the encoder; $T_s(n)$ is the time to start the encoding of the nth frame; $T_e(n)$ is the time to end the encoding of the nth frame; and $T_r(n)$ is the time to remove the coded nth frame from the BOB.

It should be noted that the inputting of the original frame S_n into the encoder as well as the occupancy of the allocated computation c_n can be regarded to be completed instantly at the moment $T_a(n)$. The movement of the coded frame with the number b_n from the encoder to the BOB is assumed to be completed instantly too. Meanwhile, the computational cost c_n consumed by the coding of this frame is added to the total computational cost at the moment $T_r(n)$. These assumptions are reasonable, because both the memory copy operation and the proposed computation resource allocation algorithm have very low overhead. Considering the other time-consuming modules, the overhead can be ignored.

As shown in Fig. 1, the scheduling of computational costs of a frame in the proposed CAM has the identical temporal phase with the scheduling of coded bits of the same frame in the coded picture buffer (CPB) of the HRD. Therefore, the proposed CAM and the traditional HRD are connected together via the temporal correspondence. In other words, the computation resource schedule and the bits resource schedule could be controlled simultaneously.

C. Adaptive Allocation of Computational Resources

The computational resources allocation based on the defined virtual buffers are performed as follows. Suppose the VCB buffer fullness is C_n when the nth frame arrives, and the required computational cost of this frame is c_n, as shown in Fig. 1(e). Then, we get:

$$\begin{cases} T_a(n) = (n-1)/\mathrm{Fr} \\ T_s(n) = T_a(n) + C_n/Cr \\ T_e(n) = T_s(n) + c_n/Cr \\ T_r(n) = T_a(n) + D \end{cases}. \tag{2}$$

In order to avoid the overflow of the VCB (i.e., red mark), we must guarantee that the removing time of the nth frame is no earlier than the ending time of encoding this frame, i.e., $T_r(n) \geq T_e(n)$. Combining with the former formulas (2), we get $c_n \leq D\cdot \mathrm{Cr} - C_n$. In order to avoid the underflow of the VCB (i.e., yellow mark), we prefer that the VCB should not be in the state of idle, i.e., $T_a(n+1) \leq T_e(n)$. Combining with the former formulas (2), we also get $c_n \geq \mathrm{Cr}/\mathrm{Fr} - C_n$. Assume that the maximum and minimum computational costs of coding the current frame are $C_{n,\max}$ and $C_{n,\min}$ respectively. Then, we get the upper bound U_n and the lower bound L_n of the VCB, i.e.,

$$\begin{cases} U_n = \min(D\cdot \mathrm{Cr} - C_n, C_{n,\max}) \\ L_n = \max(0, \mathrm{Cr}/\mathrm{Fr} - C_n, C_{n,\min}) \end{cases}. \tag{3}$$

In the practical application, $C_{n,\max}$ and $C_{n,\min}$ can be estimated based on dynamic statistic of encoding history. For example, in our experiments in Section V, we calculate:

$$C_{n,\min} = \begin{cases} k_1 \cdot C_{0,\mathrm{used}} & n = 1 \\ \min(C_{n-1,\min}, C_{n-1,\mathrm{used}}) & n \geq 2 \end{cases}, \tag{4}$$

and

$$C_{n,\max} = \begin{cases} k_2 \cdot C_{0,\mathrm{used}} & n = 1 \\ \max(C_{n-1,\max}, C_{n-1,\mathrm{used}}) & n \geq 2 \end{cases}, \tag{5}$$

where the initial values of $C_{1,\max}$ and $C_{1,\min}$ are derived from the first frame with full computational supply (i.e., without any computation constraint). We set $k_1 = 0.2$ and $k_2 = 2$ in our implementation. $C_{n,\max}$ and $C_{n,\min}$ also can be preset as constants based on offline statistic as well. Experiments show that slight difference of the initial value would not result in distinct aftereffect.

After getting the upper bounds U_n and the lower bound L_n, we can further estimate the complexity level. Assume $\mathrm{CL}_{n,\mathrm{est}}$ is the estimation of the complexity level for the current nth frame; $c_{k,\mathrm{used}}\ (\mathrm{CL}_{n,\mathrm{est}})$ is the actual computation consumption for the kth frame that is the most recently encoded with the same complexity level $(k < n)$. Then, the allocated computation for the current frame $c_{n,\mathrm{alloc}}$ is the medium among $c_{k,\mathrm{used}}(\mathrm{CL}_{n,\mathrm{est}}), U_n$ and L_n. The method to estimate $\mathrm{CL}_{n,\mathrm{est}}$ will be presented later in Section IV.B (i.e., the initial motion estimation paths decision at frame level).

In summary, the adaptive allocation of computational resource is performed hierarchically. At the frame level, we calculate the allocated computational resource $c_{n,\mathrm{alloc}}$ for the current nth frame according to the CAM by

$$c_{n,\mathrm{alloc}} = \mathrm{medium}\{\, c_{k,\mathrm{used}}(\mathrm{CL}_{n,\mathrm{est}}),\quad U_n,\quad L_n\,\}. \tag{6}$$

At the MB level, we dynamically calculate the allocated computational resource $c^i_{n,\mathrm{alloc}}$ for the current ith MB according to the available computation of current nth frame by

$$c^i_{n,\mathrm{alloc}} = \frac{c_{n,\mathrm{alloc}} - \sum_{j=0}^{i-1} c^j_{n,\mathrm{used}}}{N_{\mathrm{MB}} - i}, \tag{7}$$

where $c^j_{n,\mathrm{used}}$ is the actual computational consumption for encoding the jth MBs in current nth frame, and N_{MB} is the total number of MBs in one frame.

IV. Complexity-Configurable H.264 Encoder

A. Problem Statement

As for the utilization of the computational resources, the key point is to develop a complexity-configurable video encoder. Typically, the coding gains from different coding modules and for different contents vary. For example, the adoption of quarter-pixel motion estimation has different impacts on coding efficiency in terms of different video contents and/or different bitrates. However, the encoding complexity inevitably increases in any case [11]. In the R-D optimized coding mode decision, the complexity increases in direct proportion to the number of employed coding modes, whereas the coding gain mainly comes from a few modes [12]. Therefore, it is reasonable to allocate more computational resources to the modules that provide more coding gains. It should be noted that the relationship of complexity, rate and distortion varies in terms of video contents. To avoid using the time-consuming content analysis algorithms, we propose to decide the current coding parameters based on the coding history of previous frames.

The proposed scheme is focused on motion estimation and mode decision because they dominate the coding complexity. In order to achieve more flexibility and finer salability of complexity control, the motion estimation and mode decision modules are decomposed into a number of operational sets associated with certain complexity levels. The complexity level is decided prior to the encoding. In particular, the complexity level of motion estimation is decided at frame level, and the complexity level of mode decision is decided at macroblock level.

B. Complexity-Adjustable Motion Estimation

As we know, a number of fast motion estimation algorithms have been developed for video encoding, including the one adopted in the H.264 reference encoder [14]. Thanks to the early termination mechanism based on the zero-block detection, the complexity is reduced with little sacrifice of the R-D performance. Therefore, it is desirable to develop the CAME scheme based on the fast ME. Consequently, the largest computational costs of the CAME are not more than that of the original fast ME.

The core of the proposed CAME algorithm is the selection of the motion estimation operational path with coding performance and complexity optimization. Typically, the whole fast motion estimation process consists of two stages: integer-pixel motion search and sub-pixel motion search. The integer-pixel motion search is composed of three steps: finding an initial searching point by MV predictors; refining the initial searching point by using various searching patterns; and further refining the MV by using a small search pattern. The second step takes a large ratio of the overall computations. In some cases, this step can be skipped because the first step can achieve a MV with sufficient accuracy. In some other cases, it has to be performed, especially for the videos with complex motions and textures. The sub-pixel motion search has the similar steps and features. As a conclusion, the computational resources can be saved by skipping some unworthy steps, without the perceptible sacrifice of R-D performance. The key problem is how to select the steps to be used or skipped before they are actually performed.

Based on the above analysis, we separate the fast motion estimation in the H.264 encoder into two motion estimation operational paths with four termination points. Fig. 2 shows an exemplary set of the relations between the R-D and computational costs of the two paths. As shown in Fig. 2, each operational path includes multiple motion estimation operations, indicated by A, B, C, D and E, as defined in Table III. The motion estimation starts from A and terminates at one of B, C, D and E. The R-D cost represents the evaluated coding cost of a video frame with respect to the encoding rate and distortion during the select motion estimation operations. In general, the motion estimation operations associated with path A-B-D require less computation than the counterpart motion estimation operations associated with motion estimation operational path A-C-E. However, the former path also results in higher R-D cost than the latter one. We defined the ratio between the R-D performance gain and the computational costs increasing as:

$$\text{slope}(X - Y) = \frac{J(X) - J(Y)}{C(Y) - C(X)}, \tag{8}$$

where $J(X)$ and $C(X)$ indicate the R-D cost and computational cost with regards to the motion estimation operations X. Typically, the operational path with larger slope should be selected, because it also indicates better R-D performance gain with equal computational consumption. Then, a further problem is how to predict the slope, that is how to get $J(X)$ and $C(X)$ prior to the encoding of a video frame.

In particular, the Lagrange R-D cost function is employed as the measurement of motion estimation efficiency, which is defined as [22]:

$$J_{\text{motion}}(m, c) = \min_{\text{MV}_i \in \Omega(c)} \{\text{SAD}(\text{MV}_i, m) + \lambda \cdot R(\text{MV}_i, m)\}, \tag{9}$$

where $\Omega(\text{c})$ is the set of candidate MV_i for mode m under a restriction of available complexity level c, and λ is the Lagrange factor referring to [4] and [22]. Then, we get the total R-D costs of nth frame as:

$$J(X) = \sum_{\text{MB}_k \in \text{FRM}_n} \min_{m_j \in M(X, \text{MB}_k)} \{J_{\text{motion}}(m_j, X)\}, \tag{10}$$

where $M(X, \text{MB}_k)$ is the set of candidate mode m_j of macroblock MB_k, under the motion estimation operation X associated with the restriction of complexity level; and FRM_n is the set of macroblocks in current frame.

The computational costs of nth frame under the motion estimation operations X are calculated as:

$$C(X) = c_{\text{pro}}(X) + \sum_{\text{MB}_k \in \text{FRM}_n} \times \sum_{m_j \in M(X, \text{MB}_k)} \sum_{\text{MV}_i \in \Omega(X)} c_{\text{motion}}(\text{MV}_i, m_j, X), \tag{11}$$

where $c_{\text{motion}}(\text{MV}_i, m_j, \text{MB}_k)$ is the computational costs to process candidate motion vector MV_i of candidate mode m_j of macroblock MB_k, and it is traced for each candidate motion estimation searching point under ME operation X; $c_{\text{pro}}(X)$ is the computational costs of other encoding modules (e.g., DCT transform) during coding current frame.

The statistics show that the coding costs $J(X)$ and $C(X)$ of two successive frames are usually very close if the same motion estimation operations X are performed. Since the actual R-D costs and computational costs of the current frame are unknown prior to the encoding, the coding costs of the previous frame

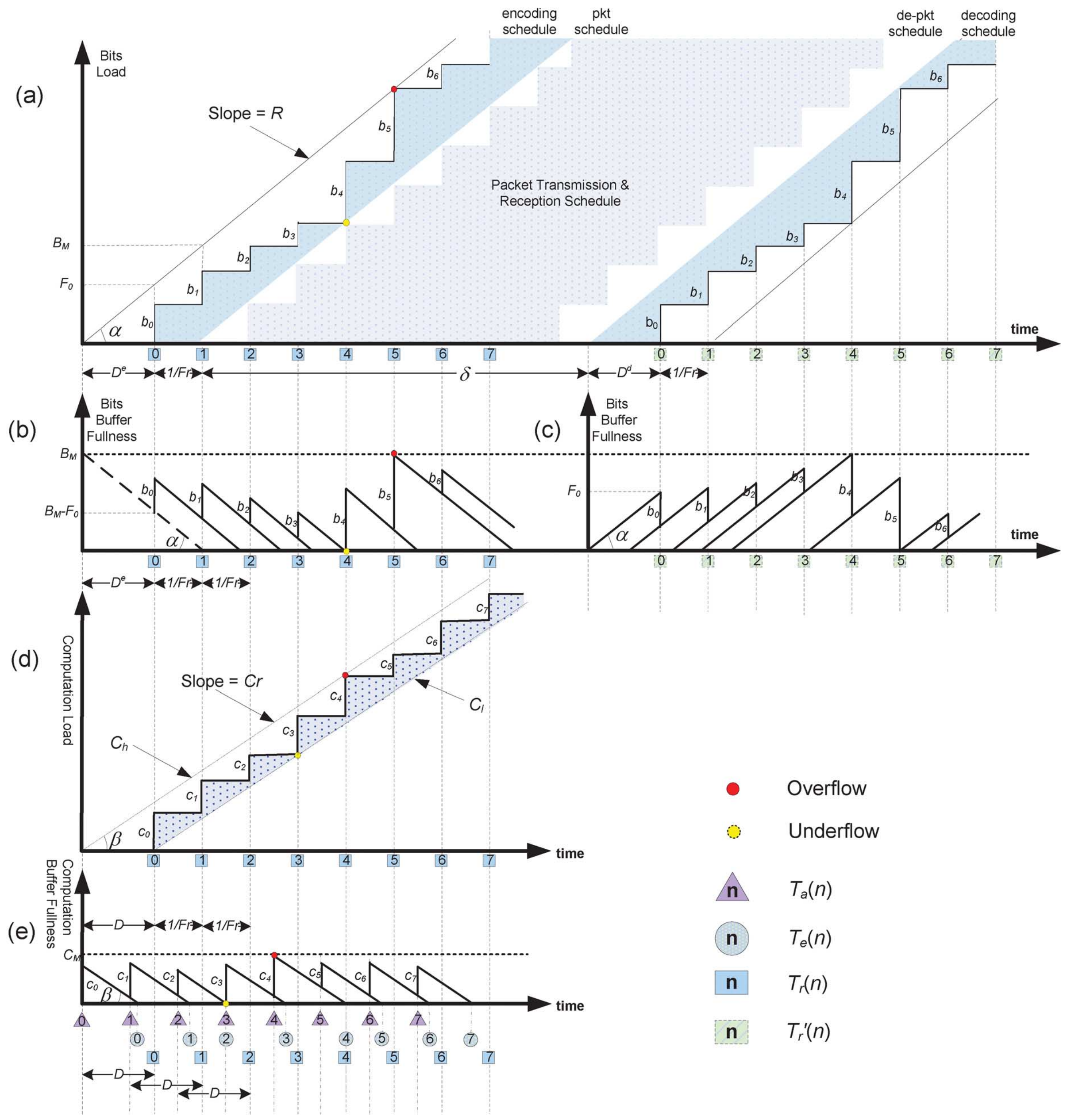


Fig. 1. The illustration of the traditional HRD and the proposed CAM. (a) HRD with leaky bucket bounds; (b) the encoder bit-stream output buffer (BOB); (c) the decoder bit-stream input buffer (BIB); (d) the virtual computation buffer of the encoder; (e) the leaky bucket bounds in the proposed CAM. (a) HRD with leaky bucket model, (b) BOB, (c) BIB, (d) CAM Leaky Bucket model, (e) VCB.

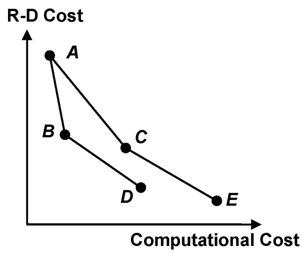


Fig. 2. Relationship between the R-D cost and the computational cost in fast ME.

can be used instead. Note that, according to the definition of the motion estimation operations X, B must be performed when the stop point is D or C, and C must be performed when the stop point is E. If the motion estimation operations X are not performed in the coding of the previous frame, $J(X)$ of the most recent frame would be used.

In the proposed scheme, our principle is to ensure optimal R-D performance within a given computation resource constraint. In theory, more computation could achieve better R-D performance. If the computation resource is sufficient enough, we targets at better R-D performance; otherwise, we jointly consider the R-D costs and computational costs to achieve a tradeoff. Consequently, we decide the motion estimation operation path (i.e., complexity level) by following two steps.

In the first step, we assume the computational resource is sufficient, and the principle is to obtain better RD performance with

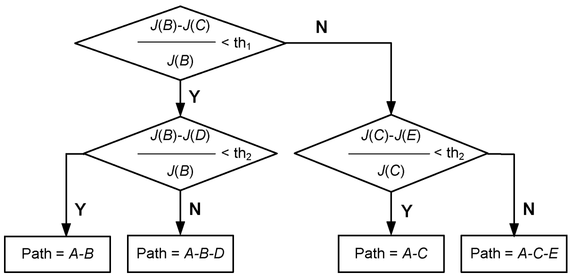


Fig. 3. Initial motion estimation operational path selection.

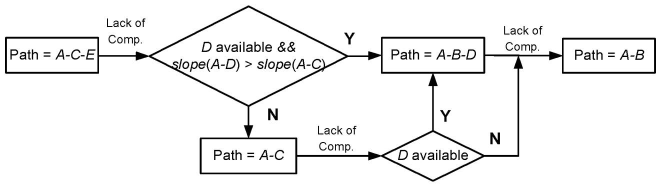


Fig. 4. Backward motion estimation operational path selection.

the computational consumption as little as possible. Therefore, we choose the path with similar RD costs but lower computational costs. In particular, according to the definition of ME operations B and C, the computational costs $C(B)$ must be lower than $C(C)$. We only need to check the $\mathrm{ratio}(J(B) - J(C))/J(B)$ to detect whether ME operations B with lower computational costs could achieve similar RD performance with ME operations C. When the ratio is less than a threshold, the path A-B-D can probably achieve the same R-D performance as the path A-C-E, whereas it has less computational cost. Further, the termination points can be determined within the selected path by the similar choosing algorithm, i.e., based on the ratio $(J(B) - J(D))/J(B)$ or $(J(C) - J(E))/J(C)$ and the threshold th_2. Based on experiential tests, we set $\mathrm{th}_1 = 0.02$ and $\mathrm{th}_2 = 0.01$ espectively. Fig. 3 shows the flowchart to decide the initial ME operational path (i.e., the $\mathrm{CL}_{n,\mathrm{est}}$ in Section III.C).

In the second step, we check if the estimated computational consumption of initial operation path would exceed the available computation resource constraint (i.e., the allocated computation $c_{n,\mathrm{alloc}}$ by the proposed CAM in Section III.C). If the available computation resource is sufficient enough, adopt the initial path from the first step. Otherwise, we decrease the complexity level by considering the slope, as shown in Fig. 4. The entrance point is one of the paths A-C-E, A-C and A-B-D, which is selected in the initial process. It flows to the next one with lower computational costs until the computation meets the constraint.

C. Complexity-Adjustable Mode Decision

The core of the proposed complexity-adaptive mode decision algorithm is the selection of the search order and termination point in the R-D optimized mode decision. According to the statistics [11], [12], the complexity increases in direct proportion to the number of modes actually being searched, while the R-D performance gain saturates for only few modes. Therefore, it is possible to reduce the computational costs without sacrificing the R-D performance by early termination after going through only a few modes in mode decision, if the actual optimal mode is among the candidates. The key problem is how to adaptively select the candidate modes as well as their orders to be gone through before an MB is actually coded.

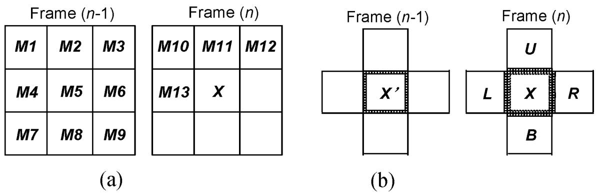


Fig. 5. Spatial and temporal neighbors of the current MB (a) Neighboring MBs, (b) Boundary Pixels.

TABLE III
COMPLEXITY LEVEL OF MOTION ESTIMATION

ME operation	Explanation
A	Without motion search (e.g., by setting MV as zero)
B	Reduced integer-pixel ME
C	Regular integer-pixel ME
D	Reduced Integer-pixel ME + sub-pixel ME
E	Regular integer-pixel ME + sub-pixel ME

TABLE IV
AVERAGE MATCH RATE OF THE FIRST PRIORITY MODE SET

Seq.\ QP	24	28	32	36	40
Akiyo	92%	96%	96%	97%	98%
Mother	67%	64%	71%	78%	85%
Salesman	71%	77%	80%	83%	88%
Foreman	53%	54%	57%	57%	60%
Coastguard	68%	73%	62%	65%	78%

To solve the above problem, we first get some statistics of the mode distribution. As we know, the mode distribution is not stable and closely related to the content and quantization parameters. Since the extraction of the content features results in large overhead in computing, the modeling of the content may not be a practical solution. Fortunately, the similarity of the R-D selected optimal coding modes exists among neighboring blocks due to the spatial and temporal correlations of a video. In particular, we classify all coding modes into 4 modes sets: *SKIP*, *Inter16* ($P16 \times 16$ to $P8 \times 16$), *Inter8* ($P8 \times 8$ to $P4 \times 4$) and *Intra* ($I16 \times 16$ and $I4 \times 4$). As shown in Fig. 5(a), the current MB X has 13 spatial and temporal neighboring MBs. For the current MB, the candidate mode sets are prioritized according to their emerging frequency in the neighboring MBs. Actually, as shown in Table IV, it can achieve over 50% match rate that the optimal coding mode is exactly among the first priority mode set. In other words, the similarity of the optimal mode among neighboring blocks is also independent of the video content and quantization parameters to a large extent. Thus, it is possible to decide the candidate modes in the R-D optimized mode decision based on coded spatial and temporal neighboring blocks.

Similar to the frame-level CAME algorithm, the proposed MB-level CAMD algorithm also involves several complexity levels. Fig. 6 shows the relationship between the R-D and computational costs, in which A–E are employed to indicate the

TABLE V
COMPLEXITY LEVEL OF MODE DECISION

MD operation	Explanations
A	RDO mode decision with copying co-located MB
B	RDO mode decision with prior one mode set
C	RDO mode decision with prior two mode sets
D	RDO mode decision with prior three mode sets
E	RDO mode decision with all mode sets

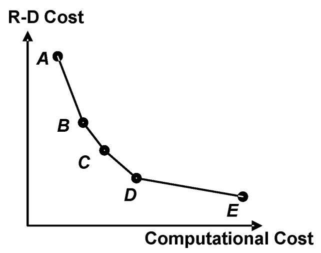


Fig. 6. Relationship between the R-D cost and the computational cost in RDO mode decision.

mode decision operations as defined in Table V. If the computational budget is not enough, the mode selection process is then terminated. What's more, with the early termination technology in fast ME, the coding mode with sufficient R-D performance might be achieved during the earlier searching process. In other words, the allocated computation for current MB might not be used up in this case.

To avoid the inaccurate mode decision that may propagate to the following frames, we always perform the full R-D optimized mode decision for the first inter frame in a group of pictures (GOP). Moreover, the spatial or temporal correlation may suddenly change due to, for example, the scene change. We take the mean absolute difference of boundary pixels (BPD), as shown in Fig. 5(b), as the spatial or temporal correlation measurement [24], [25]. If the BPD value of the current MB is over K times larger than the average in the previous frame, we take the Intra as the first candidate mode set in the mode decision process.

D. Processing Steps

In summary, the proposed complexity-constrained H.264 encoder performs the complexity adjusting as below:

Frame-Level Complexity Control:

Step 1: When encoding the first I frame or the first P frame, set the maximum complexity level of motion estimation and mode decision. Meanwhile initialize the buffer state and the boundary condition of CAM, referring to Section III.C;

Step 2: After finishing coding one frame, update the buffer sate and select the motion estimation operation path associated with complexity level $\mathrm{CL}_{n,\mathrm{est}}$, referring to Fig. 3 in Section IV.B;

Step 3: Allocate the computation resource $c_{n,\mathrm{alloc}}$ for the coming frame, referring to (3)–(6) in Section III.C;

Step 4: Select the final motion estimation operation path according to the allocated computation, referring to Fig. 4 in Section IV.B;

Step 5: Encode each MB in current frame in scan order, and meanwhile trace the actual R-D costs $J(X)$ and the actual computational costs $C(X)$ of current frame, referring to the following macroblock-level complexity control.

Macroblock-Level Complexity Control:

Step 1: Allocate the computation resource $c^i_{n,\mathrm{alloc}}$ for the current macroblock, referring to the (7) in Section III.C;

Step 2: According to the allocated computation resource $c^i_{n,\mathrm{alloc}}$, select the candidate modes and their searching order, referring to Section IV.C;

Step 3: Encode current MB according to above motion estimation and mode decision complexity level. Meanwhile trace and update the RD costs and the computational costs for each mode in current MB, referring to equations in Section IV.C.

V. EXPERIMENTAL RESULTS

We implement the proposed algorithms on the H.264 reference software JM10.2 (baseline profile) [23], and then perform the test on various video sequences. The encoding frame rate is 30 frames per second, with the GOP structure of "IP...P". With rate control algorithm in [4], the different target bit-rates are tested, including 24, 32, 64, 96 and 128 kbps. Moreover, the R-D optimization is turned on in all testing. We take the original H.264 encoding with fast motion estimation algorithm in [14] as an anchor, indicated by "Ref. S/W with FME". We then denote the overall CU number of the anchor as 100% computational costs. Note that the actual number of anchor CUs varies with different video sequences. In the tests of the proposed encoder, the target computational cost is set as a specific percentage (from 100% to 5%) of the anchor CUs.

A. Control Performance

In this section, we present the experimental results in terms of the R-D performance and the accuracy of complexity control. Figs. 7 and 8 show the R-D performances of the proposed encoder under the different complexity constraints, compared with the anchor results. When the computation consumption is the same as the anchor, the R-D performance is almost the same. When the computation consumption is reduced to 20% of the anchor, the R-D performance is only slightly lost, i.e., less than 0.2 dB in PSNR. It should be noted that, in the latter case, the proposed encoder is already as 5 times fast as the reference encoder with the fast ME. For the case of 10% computational costs, the R-D performance indeed drops more at higher bit-rate for the comparison of computation constraint with 20% and with 100%. It is mainly because the *SKIP* mode is used too frequently when the computational resources are not enough. In this case, even though the bit-rate is high and the QP might be very small, the PSNR would perceptibly drop. However, for the low bit-rate coding, the integer-pixel motion vector and *SKIP* mode would likely be selected whatever the computation resource is sufficient or limited. Thus, the PSNR dropping would not be so distinct. When the complexity level is further reduced, the R-D performance of the proposed approach also granularly changes with some quality loss, as shown in Fig. 8. However, it should be noticed that there is still no frame dropping at the

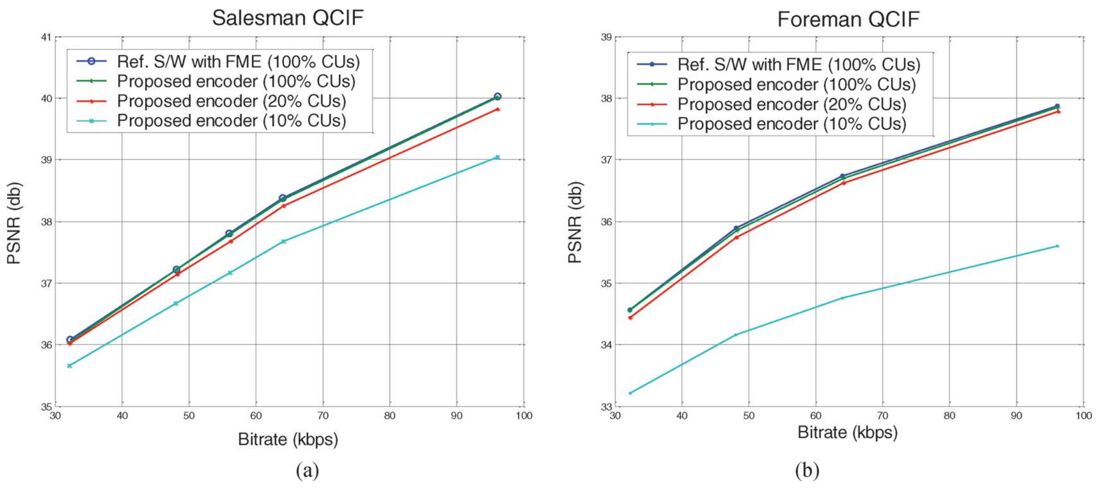


Fig. 7. Bit-rate vs. PSNR under the different computation constraints.

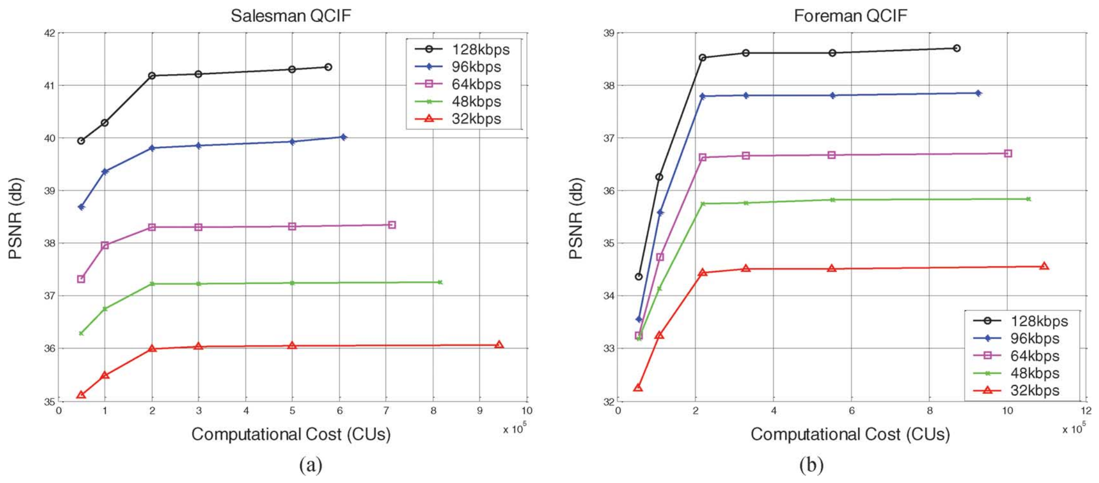


Fig. 8. Computational cost vs. PSNR under the different bit-rates.

target bit-rate and complexity level, even when the computation consumption is reduced to 5% of the anchor.

The accuracy of complexity control is evaluated as below. Fig. 9 shows the actual computational costs of each frame achieved from the proposed and reference encoders with different complexity levels. It can be observed that the computational consumption of the proposed encoder remains stable at the frame level along the time, regardless of the diversity of video contents. This feature is desirable in the environment with the constraint of maximum processor workload in addition to the constraint of overall power consumption. Although the computational consumption is stable at frame level, it varies at the MB level thanks to the proper early termination, so as to optimally utilize the computation budget. A further concern regarding the computation allocation is that the qualities of the reconstructed frames may fluctuate. Fig. 10 shows the PSNR of every frame along the time. It can be observed that the fluctuation of the PSNR from the proposed encoder has the similar trend to that from the reference encoder. In other words, the proposed encoder can adaptively achieve the desired power consumption accurately and stably, without sharp fluctuation of video quality.

B. *Comparing With DRA*

We further compare the proposed algorithm with a simple complexity control algorithm, namely, direct resource allocation (DRA). The DRA always averagely allocates the computational resources to each frame and each MB. As for the motion estimation, the original fast motion estimation algorithm [4] is performed if the allocated computation resource is sufficient; otherwise, the continued MV refining is terminated. As for the mode decision, the DRA always checks the *SKIP* mode and the $P16 \times 16$ mode first, because these two modes usually achieve the most R-D gains with less computational costs. Then, other modes in block-size order are checked until the available computation is insufficient.

Table VI shows the objective evaluation of the proposed scheme and the DRA scheme. To evaluate the visual quality, some reconstructed pictures are shown in Figs. 11 and 12. In the proposed encoder, the objective and subjective perception are both approximate when the computation reduces to 20%. By virtue of the optimal search order and search path selection, the proposed CAME and CAMD algorithms can find the optimized motion estimation path and MB modes for the different

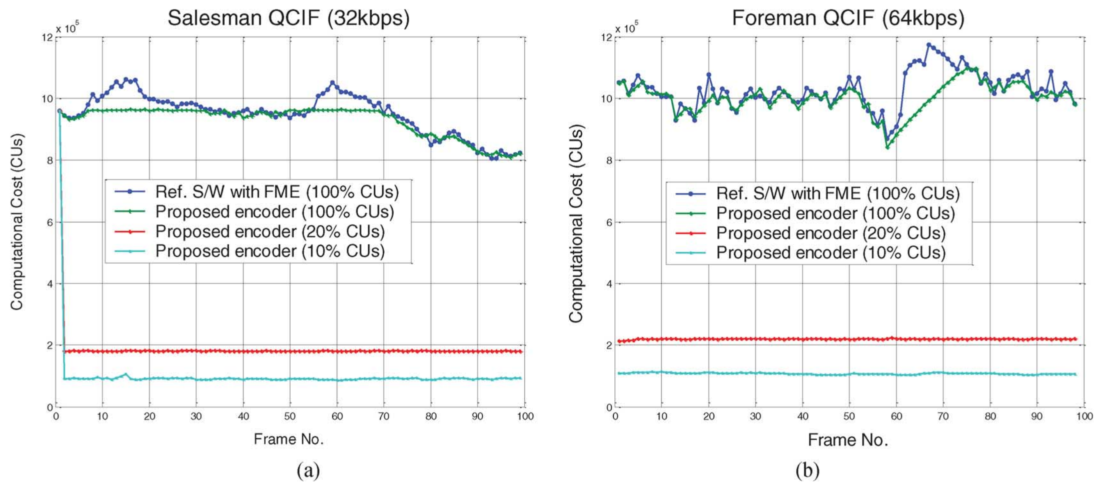


Fig. 9. Actual computational costs of 100 frames under the different computation constraints.

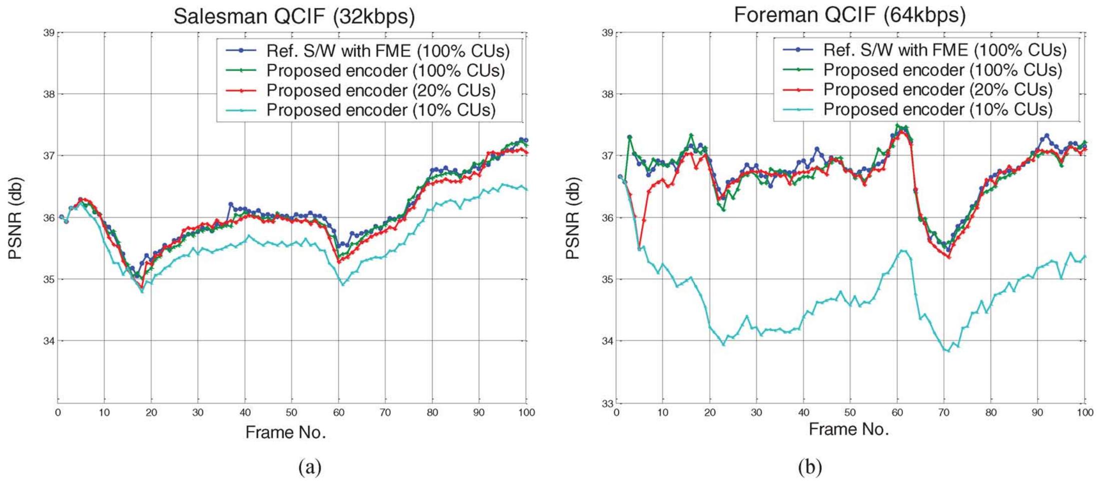


Fig. 10. PSNRs of 100 frames under the different computation constraints.

regions with limited computational resources. For example, it is desirable to code the static background with *SKIP* mode, but code the active motion regions with non-skip mode with certain motion estimation complexity level. Therefore, even when the computation reduces to 5%, the video quality from the proposed encoder is still acceptable. However, with the DRA encoder, the *SKIP* mode appears universally when the computational resources are limited, which results in quality loss obviously.

VI. Conclusion

In this paper, we have presented a joint complexity-distortion optimization approach for complexity-configurable video encoding in the environment with power constraints. The power constraints are translated to the encoding computational costs measured by the number of scaled CUs. The solved problems include the allocation and utilization of the computation resources. The allocation of restricted computational resources to each video frame is based on the proposed computation allocation model with a virtual computation buffer. The utilization of the allocated computational resources is realized with the proposed CAME and CAMD algorithms that compose a complexity-adjustable video encoder. By referring to the coding feature of previous frames, the temporal and spatial relationships are utilized to help deciding coding parameters efficiently, while avoiding the time-consuming content analysis algorithms.

The proposed algorithms can be easily incorporated into any existing H.264 encoders as well as other standard video encoders based on the hybrid coding framework. Moreover, the proposed CAM for computation scheduling and the traditional HRD for bits scheduling have the same temporal phases in operations, which makes it easier to combing the CAM and the HRD for joint complexity and rate control.

References

[1] Q. Zhang, Z. Ji, W. Zhu, and Y.-Q. Zhang, "Power-minimized bit allocation for video communication over wireless channels," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 12, pp. 398–410, June 2002.

[2] Z. He, Y. Liang, L. Chen, I. Ahmad, and D. Wu, "Power-rate-distortion analysis for wireless video communication under energy constraints," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 15, pp. 645–658, May 2005.

[3] W. Pu, Y. Lu, and F. Wu, "Joint power-distortion optimization on devices with MPEG-4 AVC/H.264 codec," in *ICC 2006*, Istanbul, Turkey, June 2006, pp. 441–446.

[4] S. Ma, W. Gao, and Y. Lu, "Rate-distortion analysis for H.264/AVC video coding and its application to rate control," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 15, no. 12, pp. 1533–1544, Dec. 2005.

[5] Y. Liang and I. Ahmad, "Adaptive techniques for simultaneous optimization of visual quality and battery power in video encoding sensors," in *Proceedings of International Conference on Image Processing*, Atlanta, GA, Oct. 2006, pp. 2477–2480.

[6] K. Deb*, Multi-Objective Optimization Using Evolutionary Algorithms*. : Wiley, 2001.

[7] S. M. Akramullah, I. Ahmad, and M. L. Liou, "Optimization of H.263 video encoding using a single processor computer: Performance tradeoffs and benchmarking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 8, pp. 901–915, Aug. 2001.

[8] Y. Eisenberg, C. E. Luna, T. N. Pappas, R. Berry, and A. K. Katsaggelos, "Joint source coding and transmission power management for energy efficient wireless video communications," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 12, no. 6, pp. 411–424, June 2002.

[9] P. Agrawal, J.-C. Chen, S. Kishore, P. Ramanathan, and K. Sivalingam, "Battery power sensitive video processing in wireless networks," in *The Ninth IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, Mar. 1998, vol. 1, pp. 116–120.

[10] D. Li, Y. Sun, and Z. Feng, "Joint power allocation and rate control for real-time video transmission over wireless system," *IEEE GLOBECOM 2005*, vol. 4, p. 5, Nov. 2005.

[11] T. Wiegand, G. J. Sullivan, G. Bjntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, July 2003.

[12] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammar, and T. Wedi, "Video coding with H.264/AVC: Tools, performance, and complexity," *IEEE Circuits and System Magazine*, vol. 4, no. 1, pp. 7–28, 2004.

[13] S. Saponara, C. Blanch, K. Denolf, and J. Bormans, "The JVT advanced video coding standard: Complexity and performance analysis on a tool-by-tool basis," in *Packet Video Workshop (PV'03)*, Nantes, France, Apr. 2003.

[14] Z. Chen, P. Zhu, and Y. He, "Fast integer pel and fractional pel motion estimation for JVT," in *JVT-F017, JVT 6th Meeting*, Awaji, JP, Island, Dec. 2002.

[15] B. Jeon and J. Lee, "Fast mode decision for H.264," in *JVT-J033, JVT 10th Meeting*, Waikoloa, Hawaii, USA, Dec. 2003.

[16] Y.-H. Kim, J.-W. Yoo, S.-W. Lee, J. Shin, J. Paik, and H.-K. Jung, "Adaptive mode decision for H.264 encoder," *Electronics Letters, 16th*, vol. 40, no. 19, Sept. 2004.

[17] Y. Liang, I. Ahmad, and J. Luo, "Joint power and distortion control in video coding," *SPIE Video Communications and Image Processing*, vol. 5685, pp. 885–895, 2005.

[18] C. S. Kannangara, I. E. G. Richardson, M. Bystrom, J. Solera, Y. Zhao, A. MacLennan, and R. Cooney, "Low-complexity skip prediction for H.264 through lagrangian cost estimation," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 16, no. 2, pp. 202–208, Feb. 2006.

[19] T. D. Burd, T. A. Pering, A. J. Stratakos, and R. W. Brodersen, "A dynamic voltage scaling microprocessor system," *IEEE Journal of Thereforlid-State Circuits*, vol. 35, no. 11, pp. 1571–1580, Nov. 2000.

[20] T. Burd and R. Broderson, "Processor design for portable systems," *The Journal of VLSI Signal Processing*, vol. 13, no. 2, pp. 203–222, Aug. 1996.

[21] R.-C. Jordi, P. A. Chou, and S. L. Regunatha, "A generalized hypothetical reference decoder for H.264/AVC," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 674–687, July 2003.

[22] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 74–90, Nov. 1998.

[23] , [Online]. Available: http://iphome.hhi.de/suehring/tml/download/

[24] L. Su, Y. Zhang, W. Gao, Q. Huang, and Y. Lu, "Improved error concealment algorithms based on H.264/AVC non-normative decoder," *IEEE ICME 2004*, vol. 3, pp. 1671–1674, 2004.

[25] V. Varsa, M. M. Hannuksela, and Y. K. Wang, "Non-normative error concealment algorithms," in *VCEG-N62, ITU 14th Meeting*, Santa Barbara, CA, USA, Sept. 2001.

Li Su received the B.S. degree in computer science from Xiangtan University, Xiangtan, China, in 2002. She is currently pursuing the Ph.D. degree in computer science from Graduate School of the Chinese Academy of Sciences.

From 2006 to 2007, she was a visiting student with Microsoft Research Asia, Beijing, China. Her research interests include video compression and robust video communication.

Yan Lu (S'02-M'07) received the B.S., M.S. and Ph.D. degrees from Harbin Institute of Technology (HIT), Harbin, China, in 1997, 1999 and 2003, respectively, all in computer science.

He was a Research Assistant with the Department of Computer Science, City University of Hong Kong, Hong Kong SAR, during 1999 to 2000. He was with the Joint R&D Lab (JDL) for advanced computing and communication, Chinese Academy of Sciences, Beijing, China, during 2001 to 2004. Since April 2004, he has been with Microsoft Research Asia, where he is now a Researcher. His research interests include image and video coding, multimedia streaming, and compression-enabled graphics applications.

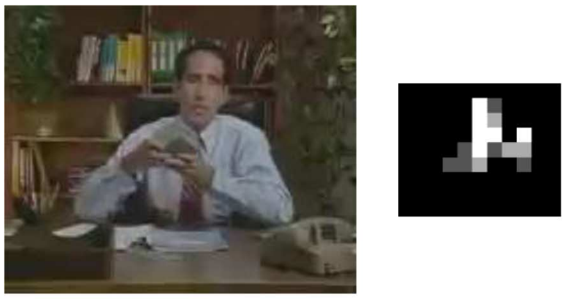
Feng Wu (M'99-SM'06) received the B.S. degree in electrical engineering from XIDIAN University in 1992. He received the M.S. and Ph.D. degrees in computer science from Harbin Institute of Technology in 1996 and 1999, respectively.

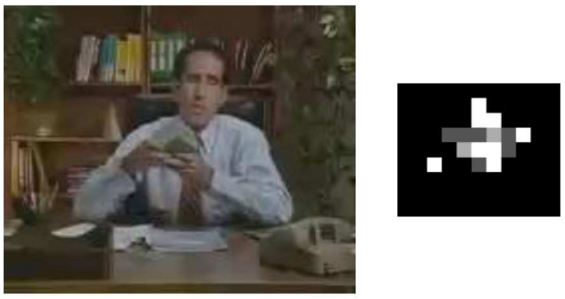
He joined in Microsoft Research China as an associated researcher in 1999. He has been a researcher with Microsoft Research Asia since 2001. His research interests include image and video representation, media compression and communication, computer vision and graphics. He has been an active contributor to ISO/MPEG and ITU-T standards. Some techniques have been adopted by MPEG-4 FGS, H.264/MPEG-4 AVC and the coming H.264 SVC standard. He served as the chairman of China AVS video group in 2002–2004 and led the efforts on developing China AVS video standard 1.0. He has authored or co-authored over 100 conference and journal papers. He has about 30 U.S. patents granted or pending in video and image coding.

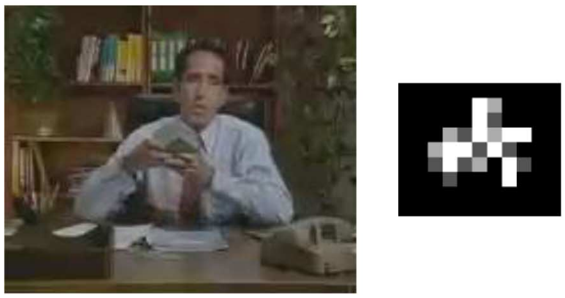
Shipeng Li (M'97) received the B.S. and M.S. degrees from the University of Science and Technology of China (USTC), Hefei, in 1988 and 1991, respectively, and the Ph.D. degree from Lehigh University, Bethlehem, PA, in 1996, all in electrical engineering.

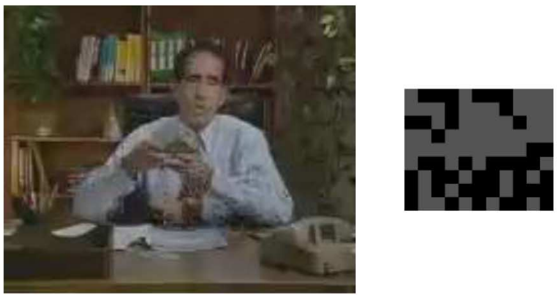
He was with the Electrical Engineering Department, USTC, during 1991–1992. He was a Member of Technical Staff at Sarnoff Corporation, Princeton, NJ, during 1996–1999. He has been a Researcher with Microsoft Research Asia, Beijing, China, since May 1999, and is now a Principal Researcher and Research Manager. His research interests include image/video compression and communications, digital television, wireless and mobile communication, and digital rights management and security. He has contributed several technologies to the MPEG-4 and H.264 international standards.

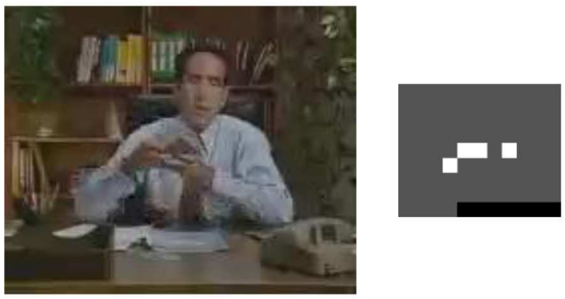
Dr. Li is a member of Visual Signal Processing and Communications Technical Committee of IEEE Circuits and Systems Society and a member of Multimedia Signal Processing Technical Committee of IEEE Signal Processing Society. He serves in the Editorial Boards of IEEE Transaction on Circuits and Systems for Video Technology and Journal of Visual Communications and Image Representation. He was Special Session Chair of IEEE PCM 2000 and Local Chair of IEEE PCM 2001, the Technical Program Co-Chair for VCIP 2005, General Co-Chair of PV 2006, and a Track Co-Chair of IEEE ICME 2006.


(a)


(b)


(c)


(d)


(e)

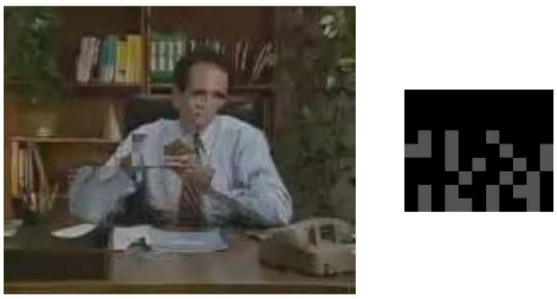

(f)

Fig. 11. The 25th frame of Salesman at 32 kbps (Left: reconstructed picture; Right: mode map with the lighter color indicating larger computational costs, i.e., black indicating SKIP mode.) (a) Ref. S/W w/FME, 100% CUs, (b) Proposed encoder, 100% CUs, (c) Proposed encoder, 20% CUs, (d) DRA encoder, 20% CUs, (e) Proposed Encoder, 5% CUs, (f) DRA encoder, 5% CUs.

Wen Gao (M'92-SM'05) received M.S. degree in computer science from Harbin Institute of Technology in 1985, and Ph.D. degree in electronics engineering from the University of Tokyo in 1991. He was a Professor in computer science at Harbin Institute of Technology from 1991 to 1995, a Professor in computer science at Institute of Computing Technology of Chinese Academy of Sciences from 1996 to 2005.

He is currently a Professor at the School of Electronics Engineering and Computer Science, Peking University, China. He has been leading research efforts to develop systems and technologies for video coding, face recognition, sign language recognition and synthesis, and multimedia retrieval. He earned many awards include five national awards by research achievements and activities. He did many services to academic society, such as general co-chair of IEEE ICME07, and the head of Chinese delegation to the Moving Picture Expert Group (MPEG) of International Standard Organization (ISO) since 1997, he is also the chairman of the working group responsible for setting a national Audio Video coding Standard (AVS) for China. He published four books and over 500 technical articles in refereed journals and proceedings in the areas of signal processing, image and video communication, computer vision, multimodal interface, pattern recognition, and bioinformatics.

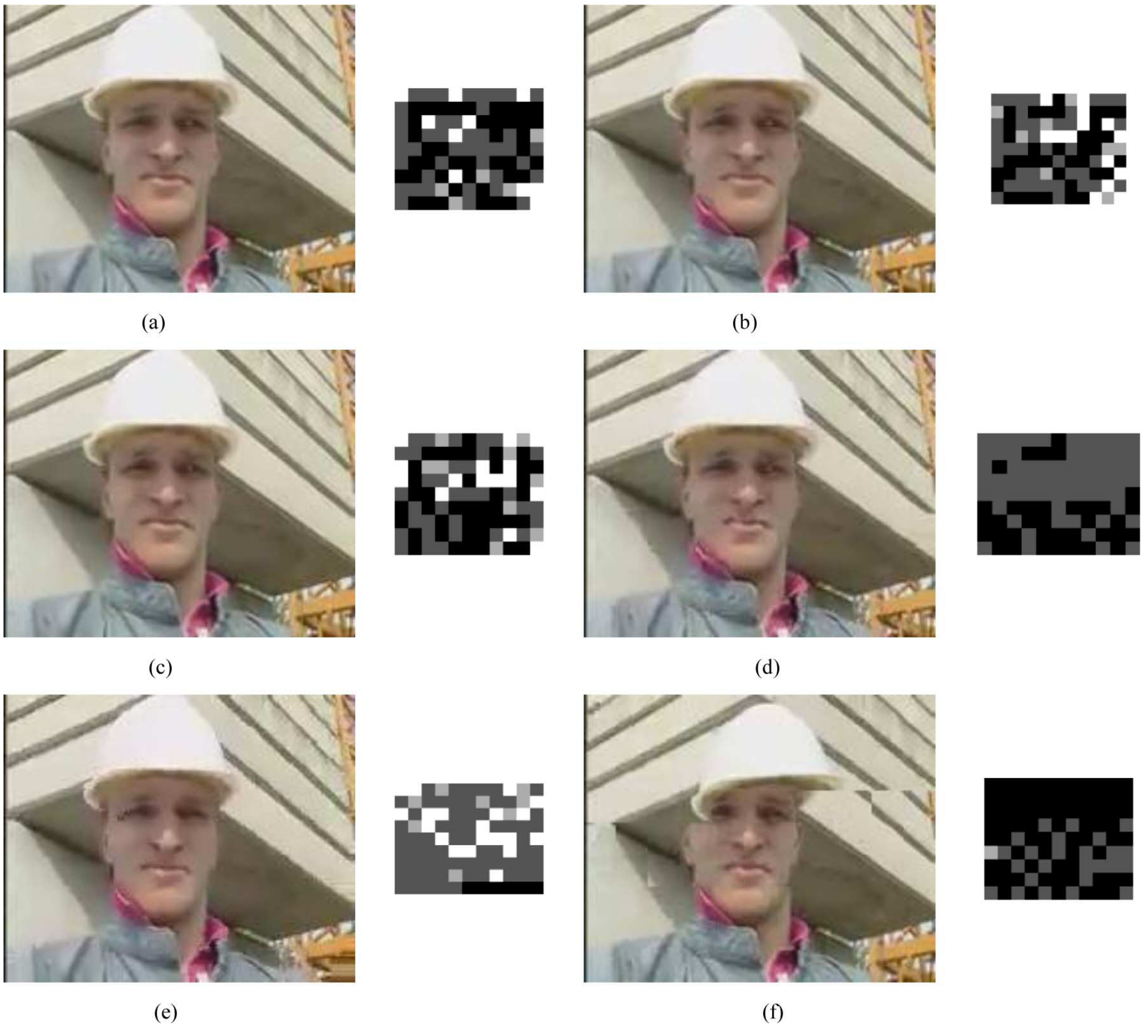


Fig. 12. The 25th frame of Foreman at 64 kbps. (Left: reconstructed picture; Right: mode map with the lighter color indicating larger computational costs, i.e., black indicating *SKIP* mode.) (a) Ref. S/W w/FME, 100% CUs, (b) Proposed encoder, 100% CUs, (c) Proposed encoder, 20% CUs, (d) DRA encoder, 20% CUs, (e) Proposed encoder, 5% CUs, (f) DRA encoder, 5% CUs.

TABLE VI
CODING PERFORMANCES OF THE PROPOSED SCHEME AND THE DRA SCHEME

Seq.	Bit-rate (bps)	Ref. S/W w/ FME		Proposed – PSNR (dB)				DRA – PSNR (dB)			
		CUs	PSNR	100%	20%	10%	5%	100%	20%	10%	5%
Akiyo	24k	714135	39.01	39.02	38.92	38.38	36.85	38.97	37.93	36.75	35.4
	32k	642994	41.36	39.95	39.86	39.08	37.64	40.34	39.37	38.15	36.45
	48k	601816	42.15	41.68	41.55	40.7	39.25	41.82	41.16	39.61	37.33
Sales-man	32k	948845	36.08	36.07	35.99	35.48	35.11	36.01	35.09	35.04	34.41
	48k	831200	37.21	37.22	37.14	36.75	36.29	37.13	36.24	35.51	35.05
	64k	733222	38.38	38.35	38.3	37.95	37.32	38.26	37.27	36.23	35.75
Silent	48k	954152	35.93	35.91	35.77	35.41	34.70	35.84	34.39	33.96	33.05
	64k	852820	37.03	36.98	36.84	36.48	35.42	36.95	35.02	34.40	33.67
	96k	739210	38.64	38.58	38.57	37.98	36.77	38.47	37.01	35.42	34.28
Fore-man	64k	1027738	36.74	36.70	36.62	34.73	33.25	36.62	34.22	32.64	29.83
	96k	949944	37.87	37.85	37.78	35.58	33.56	37.76	35.1	33.26	30.19
	128k	901134	38.71	38.70	38.51	36.25	34.36	38.59	35.7	34.12	30.64