Computation-Aware Intra-Mode Decision for H.264 Coding and Transcoding

Jhih-Shen Shen, Chih-Hung Chen, and Chia-Ming Tsai
Department of Computer Science & Information Engineering
National Chung Cheng University
Chiayi 621, Taiwan
tsaicm@cs.ccu.edu.tw

Chia-Wen Lin
Department of Electric Engineering
National Tsing Hua University
Hsinchu 30013, Taiwan
cwlin@ee.nthu.edu.tw

***Abstract*—Nowadays, most video-enabled mobile terminals have been equipped with modern video codecs. Video communications, especially for encoding H.264 format bit-stream, however, are usually very power-consuming, leading to rather limited communication period for mobile devices powered by batteries. Computation-aware video coding can effectively extend the battery life. In this paper, we propose a computation-aware intra mode decision for H.264 coding and transcoding applications. The proposed algorithm optimizes the visual quality by adaptively adjusting the number of prediction modes in mode decision under a given computation constraint. We introduce a new concept of computation buffer and formulate the computation control of mode decision as a rate-distortion optimization problem of computation buffer control. Experimental results show that our proposed algorithm can effectively control the computational complexity while maintaining good RD-performance and satisfying the given computation constraint.**

I. Introduction

The explosive growth of compressed video streams and repositories which are accessible worldwide and the recent addition of new video-related standards, such as H.264/AVC [1] and MPEG-21, have stimulated research for new technologies and applications in the area of multimedia architectures, processing, and networking. Users may employ heterogeneous video-enabled terminals such as computers, TVs, mobile phones and personal digital assistants with a wide range of computational and display capabilities, energy resources, features, accessibilities, and user preferences. In recently years, mobile devices have been widely deployed. These mobile devices usually have rather limited battery power lifetime. Nowadays, most video-enabled mobile terminals have been equipped with modern video codecs. Video communications, especially for encoding H.264 format bit-stream, however, are usually very power-consuming, leading to rather limited communication period for mobile devices powered by batteries. How to use the limited power more efficiently for optimal video encoding thus becomes an important issue.

H.264 achieves significant improvement on rate-distortion performance by exploiting advanced video coding technologies, such as Variable Size Block Motion Estimation (VSBME), Multiple Reference Prediction, Intra Prediction, Improved Loop Filter, and Context-Based Adaptive Binary Arithmetic Coding (CABAC). It can save up to 50% in bit-rates as compared to MPEG-4 Advanced Simple Profile (ASP). The outstanding coding performance of H.264/AVC, however, comes with a cost of high complexity, making it too complex to be used for devices with limited computing resource, especially for mobile devices with limited battery power.

The issues about power consumption reduction and effective power allocation for handheld devices have been addressed in the literature. In [2] and [3], Kannangara *et al.* proposed a variable complexity algorithm for H.264, and adapted for a per-frame computational control algorithm. This computational control algorithm is based on minimizing the Lagrange Rate-Distortion-Complexity cost of the encoder. In [4], Wang *et al.* proposed a complexity adaptive motion estimation and mode decision (CAMED) method for an H.264 encoder. By giving the bit rate and computational constraints, CAMED explores the trade-off between video quality and computation resource consumption to determine the optimal motion vectors and block modes used in the motion-compensation process in the decoder.

Computation-aware video coding can effectively extend the battery life of mobile devices. In this paper, we propose a computation-aware intra mode decision for H.264 coding and transcoding applications. The proposed algorithm optimizes the visual quality by adaptively adjusting the number of prediction modes in mode decision under a given computation constraint. We introduce a new concept of computation buffer and formulate the computation control of mode decision as a rate-distortion optimization problem of computation buffer control.

II. PROPOSED COMPUTATION CONTROL ALGORITHM

H.264 performs intra prediction by using the spatial correlation with adjacent previously encoded blocks. As illustrated in Figure 2(a), *a* to *p* are the luma pixels to be predicted, and *A* to *M* are the boundary luma pixels of previously encoded adjacent blocks. Figure 1(b) shows nine prediction modes in Intra_4×4 prediction: eight directional modes and one non-directional mode (Mode2). Before encoding each 4×4 luma block, nine 4×4 prediction blocks are produced by using Intra_4×4 prediction. For example, in Mode 0, pixel *A* is used to predict the pixels *a*, *e*, *i*, and *m*; pixel *B* is used to predict the pixels *b*, *f*, *j*, and *n*, and so on. Likewise in the case of Mode 1, pixel *I* is used to predict the pixels *a*, *b*, *c*, and *d*, and so on. For Mode 3 to 8, the pixels are predicted using a directional weighted average of *A* to *M*. For example, in Mode 4, *d* is predicted by *round(B/4 + C/2 + D/4)*. In Mode 2 (DC prediction), the average of *A* to *L* is used to predicted *a* to *p*.

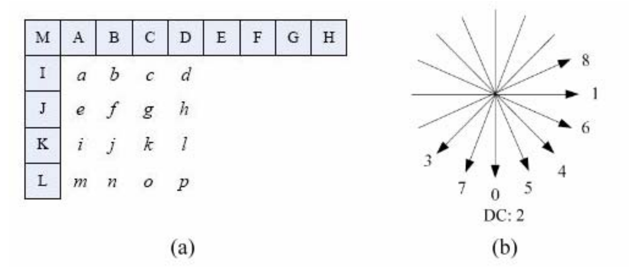


Figure 1. Intra_4×4 prediction: (a) indices for 4×4 block and adjacent pixels, (b) directions for each prediction mode.

In H.264, if all frames are intra-coded, the main computational bottleneck is the number of Intra_4×4 candidate modes for RDO computation. Our algorithm is aimed at dynamically controlling the encoding complexity subject to a given computing power constraint (i.e., the remaining battery power budget) while maintaining the video quality. We define the computation constraint as the number of prediction modes to be used per 4×4 intra block. For example, a "44%" computation constraint means that, on average four out of nine prediction modes will be chosen per 4×4 intra block in RDO computation of mode decision. With the proposed algorithm, an encoder can adaptively determine the number of candidate modes for each 4×4 intra block according to the block's characteristics under the given computation constraint. The proposed algorithm is divided into four steps as elaborated below.

A. *Computation Buffer Initialization*

Before encoding a video sequence, a computation buffer is set up based on a given computing power constraint. The computation buffer records and updates the available computing power budget up to the current block. Two control parameters are used:

1) *C*: the number of remaining non-encoded 4×4 intra blocks of the video seq*uence.*

2) *Buffer*: the number of avalible candidate modes that can be chosen for RDO computation.

For instance, given "70%" computing power constraint for a QCIF format sequence with 50 I-frames, two control parameters are determined as below:

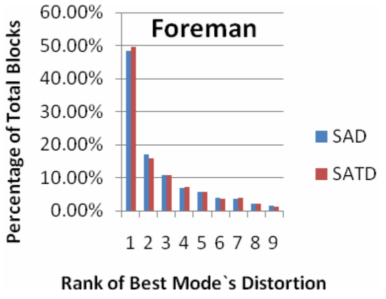
$$C = \frac{176}{4} \times \frac{144}{4} \times 50 = 79200,$$

$$Buffer = C \times 9 \times 70\% = 498960$$

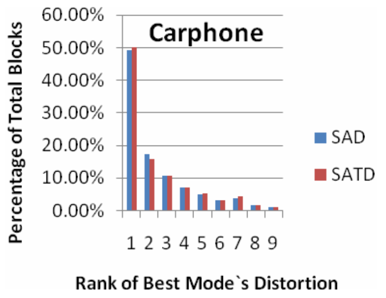
B. *Modeling*

The proposed computation model is used to determine how many candidate modes can be used for predicting each 4×4 intra-block based on the features of the block. In our model, an intra-block is characterized based on the following three observations.

1) Observation 1: the rank of best mode's prediction error. RDO computation, rather than the prediction error of prediction modes, has been used for determining the best intra-mode in the H.264 reference software for sake of accuracy. The RDO computation, however, consumes heavy computing power. To show the relationship between the best mode obtained by RDO and the SAD (Sum of Absolute Difference) or SATD (Sum of Absolute Transformed Difference) prediction errors, the SAD/SATD prediction errors of nine prediction modes are sorted by the error values. Figure 2 (a)-(b) shows the percentages in population of the best mode (obtained by mode decision with RDO) sorted by its rank of SAD and SATD prediction errors among the nine prediction modes. From the figure, we observed that the top four prediction modes (with lowest SAD/SATD values) make up more than 90% of the population. Such high percentages imply that SAD or SATD prediction error can be used to predict the best mode efficiently without significant reduction in accuracy. In our method, SATD is adopted to calculate the prediction error as it is more accurate than SAD in predicting the best coding mode.



(a)



(b)

Figure 2. Average percentages of the prediction modes that are ranked by their prediction errors being chosen as the best coding mode after mode decision with RDO for four test sequences: (a) *Foreman* and (b) *Carphone.*

2) Observation 2: the relationship between the standard deviation of SATD and the rank.

The standard deviation σ of nine prediction modes' SATD values can be obtained from (1):

$$\sigma = \sqrt{\frac{1}{9}\sum_{k=0}^{8}(SATD_k - \overline{SATD})^2}, \, and \; \overline{SATD} = \frac{1}{9}\sum_{i=0}^{8} SATD_k \qquad (1)$$

where *k* is the index of prediction modes.

These statistics can be used to characterize the relationship between the standard deviation of SATD prediction errors and the average rank of best mode SATD value. For example, if we find that the average rank of the best mode SATD is 4 while the standard deviation of SATD prediction errors is 50, the best mode can be found from the four top-ranking modes (sorted by

SATD) with a high probability if the standard deviation of SATD prediction errors is 50. With such models, we are able to use the statistics of SATD prediction errors of each block to estimate the average rank of best mode's SATD value without resorting to RDO computation, leading to significant computation reduction.

We use the linear regression to find the mathematical model to approximate the relationship curve as follows.

$$\begin{cases} Optimal_i = a + b \cdot \sigma, & for\ \sigma = 0 \\ Optimal_i = c + d \cdot \ln \sigma, & for\ \sigma > 0 \end{cases} \tag{2}$$

where $Optimal_i$ denotes the optimal number of candidate modes. The mode parameter sets (*a*,*b*) and (*c*,*d*) are determined using the least squares method. Table I lists the derived coefficient set (*a*,*b*) and (*a*,*b*) by using (3) in the *Foreman* and *Carphone* sequences.

TABLE I. The coefficient sets for 2 different test sequences with four different quantization parameters.

Sequence	QP	*a*	*b*	*c*	*d*
Foreman	28	2.3688	0.9272	3.6651	-0.3942
	32	2.3493	0.9411	4.2650	-0.4981
	36	2.1755	1.0469	5.0111	-0.6179
	40	2.0805	1.0266	5.6999	-0.7094
Carphone	28	2.1590	0.9037	3.3406	-0.3243
	32	2.1271	1.0335	3.7453	-0.3882
	36	2.0500	1.0230	4.1852	-0.4503
	40	2.0310	1.0434	5.0347	-0.5799

Figure 3 compares the actual data obtained from our experiment and the corresponding curve approximated by the proposed model in (2). The dark blue lines denote the actual data curve, whereas the red lines represent the approximated model curve. We define this curve as a "Standard Deviation-to-Rank" (SR) curve. The result shows that the proposed model is accurate enough.

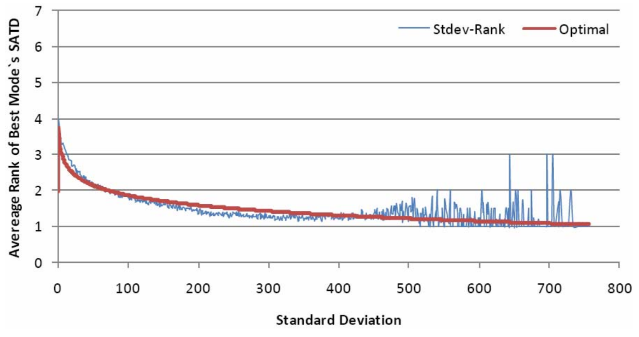


Figure 3. Comparison between the actual data and the proposed model (*Foreman*, QCIF, 300 I-Frames, QP = 18).

However, for each frame, the SR data of a video frame can be obtained after the frame has been encoded. This is only feasible for non-realtime coding and transcoding applications. In real-time applications, only the data of previous encoded frames are available, whereas the SR data of the current frame is unknown before performing RDO. However, we will show that the SR model of current frame can be estimated using the data of previous coded frames with fairly good accuracy according to the following observation.

3) *Observation 3:* Temporal similarity of SR models.

Suppose the SR data of the *j*th frame is denoted as SR_j. And the linear regression is defined as *Linest*(). The model coefficients of the *j*th frame is thus *Linest*(SR_j). The cumulative SR data of previous *t* frames, $SR_{j,t}$, is defined in (3).

$$SR_{j,t} = \sum_{i=j-1}^{j-t} SR_i \tag{3}$$

Moreover, the curve coefficients of pervious *t* frames is denoted as *Linest*($SR_{j,t}$). The sum of coefficient difference, CD_t, is defined as follows.

$$CD_t = \sum_{i=1}^{N} \left\| Linest(SR_{i,t}) - Linest(SR_i) \right\|, \tag{4}$$

where *N* is the total number of frames. As a special case, the SR data from all frames is defined as SR_{All} by using (5).

$$SR_{All} = \sum_{i=1}^{N} SR_i \tag{5}$$

The model parameter set of all frames is denoted as *Linest*(SR_{All}). Then the sum of coefficient difference, CD_{All}, is defined in (6).

$$CD_{All} = \sum_{i=1}^{N} \left\| Linest(SR_{All}) - Linest(SR_i) \right\| \tag{6}$$

By using the features defined in (3), (4), (5), and (6), the temporal similarity of SR data between current frame and previous frames can be characterized. Table II shows the coefficient difference with different values of *t*. We can observe that the coefficient difference is small when *t* is small enough, meaning that there exists temporal similarity between the SR models of current frame and previous frames. As a result, the SR curve of current frames from can be estimated form those of previous frames before encoding current frame with high reliability.

TABLE II. Coefficient Difference for Foreman QCIF

QP	CD_1		CD_5		CD_{All}	
	C	D	C	D	C	D
20	75.28	14.66	57.80	12.41	127.45	28.73
24	60.53	11.59	49.13	10.62	118.96	27.01
28	63.79	12.24	53.43	11.69	128.79	29.93
32	82.21	15.28	67.41	14.12	140.66	31.88
36	85.42	16.00	62.09	12.47	138.85	29.49
40	86.98	17.84	63.44	13.93	87.27	18.61

C. *Computation Allocation*

The objective of computation resource allocation is to determine the number of candidate modes for each 4×4 intra blocks. The computation budget for the *j*th frame, $Budget_j$, is defined as follows:

$$Budget_j = \frac{Buffer}{C}. \tag{7}$$

Before encoding a 4×4 intra block, we define two control parameters as below:

1) Budget: the computation budget for the *i*th 4×4 intra block.

2) Extra: After encoding *i*th 4×4 intra block, this control parameter is used to record the surplus computational budget for encoding next block.

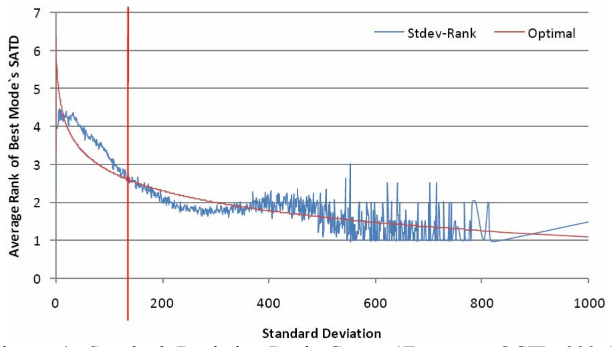


Figure 4. Standard Deviation-Rank Curve (*Foreman*, QCIF, 300 I-Frames, QP=40).

While initializing $Budget_j$, a check of standard deviation is performed to find out additional computation according to the values of *Extra*. The SR model usually drops sharply at the small standard deviation region as shown in Figure 4, that is, the SR curve value is smaller than the statistical data at left region of the red vertical line. Therefore, if the standard deviation of the *i*th block is smaller than the threshold (TH_σ) and there are some extra computations, the $Budget_i$ can be increased. The value of TH_σ is determined empirically.

Finally, the initialization of $Budget_i$ is described in (8).

$$\begin{cases} Budget_i = Budget_j + 1 \\ \quad Extra = Extra - 1 \end{cases}, \text{when } \sigma < TH_\sigma \text{ and } Extra > 1$$
$$\begin{cases} Budget_i = Budget_j \\ Extra \text{ is not change} \end{cases}, \text{otherwise} \qquad (8)$$

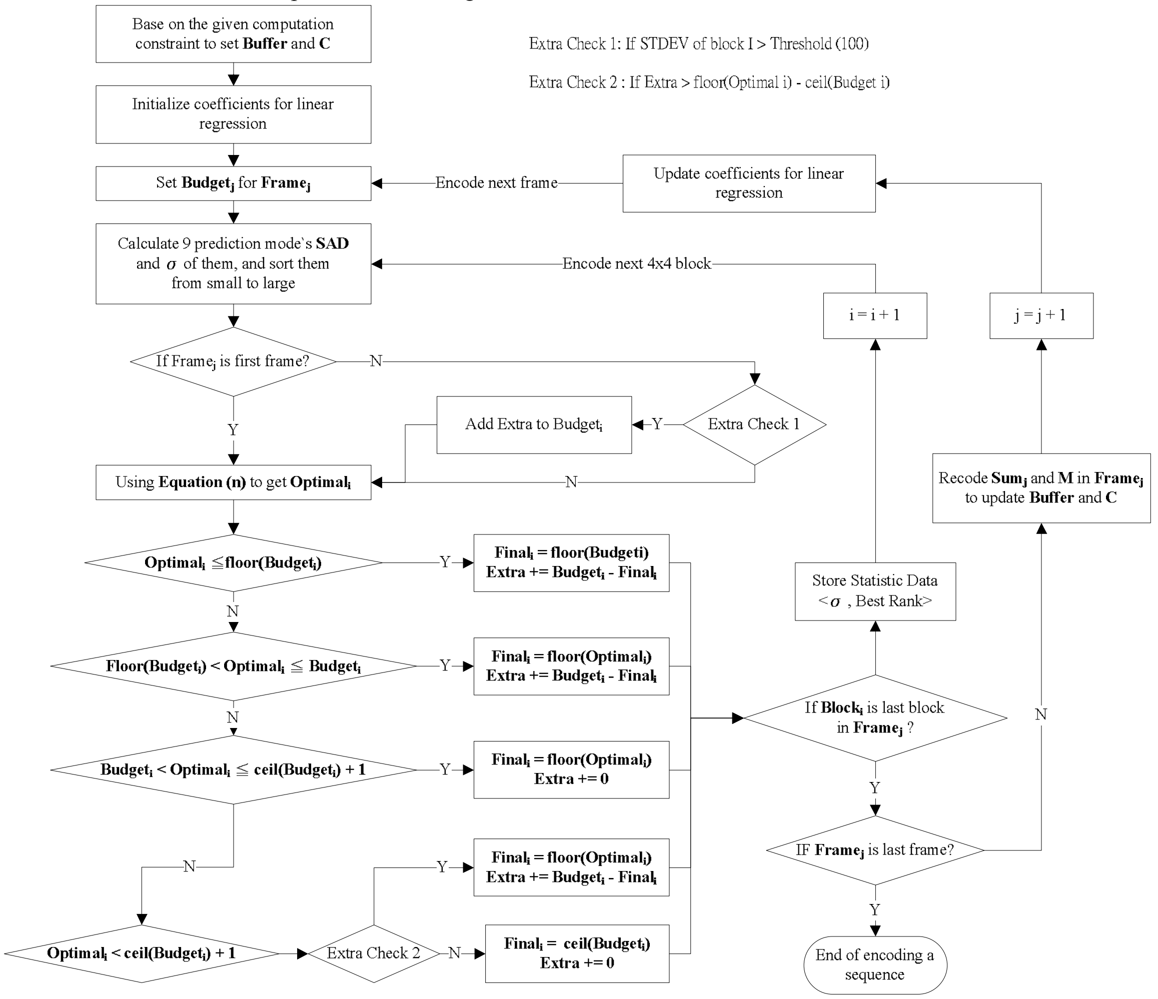

Base on the given computation constraint to set Buffer and C
Initialize coefficients for linear regression
Set Budget$_j$ for Frame$_j$
Calculate 9 prediction mode`s SAD and σ of them, and sort them from small to large
If Frame$_j$ is first frame?
Y
N
Extra Check 1
Y
N
Add Extra to Budget$_i$
Using Equation (n) to get Optimal$_i$
Optimal$_i$ ≦ floor(Budget$_i$)
Final$_i$ = floor(Budgeti) Extra += Budget$_i$ - Final$_i$
Floor(Budget$_i$) < Optimal$_i$ ≦ Budget$_i$
Final$_i$ = floor(Optimal$_i$) Extra += Budget$_i$ - Final$_i$
Budget$_i$ < Optimal$_i$ ≦ ceil(Budget$_i$) + 1
Final$_i$ = floor(Optimal$_i$) Extra += 0
Optimal$_i$ < ceil(Budget$_i$) + 1
Extra Check 2
Final$_i$ = floor(Optimal$_i$) Extra += Budget$_i$ - Final$_i$
Final$_i$ = ceil(Budget$_i$) Extra += 0
Extra Check 1: If STDEV of block I > Threshold (100)
Extra Check 2 : If Extra > floor(Optimal i) - ceil(Budget i)
Encode next frame
Update coefficients for linear regression
Encode next 4x4 block
i = i + 1
j = j + 1
Recode Sum$_j$ and M in Frame$_j$ to update Buffer and C
Store Statistic Data <σ , Best Rank>
If Block$_i$ is last block in Frame$_j$?
IF Frame$_j$ is last frame?
End of encoding a sequence

Figure 5. Flowchart of the Proposed Method.

Before choosing the candidate modes for each block, as mentioned previously, the optimal number of candidate modes can be estimated based on the features of each block. Thus, in the following, *Budget$_i$* or *Optimal$_i$* are chosen as the final number of candidate modes based on the relationship between *Budget$_i$* or *Optimal$_i$*. There are four cases of the relationship between *Budget$_i$* and *Optimal$_i$*. The final number of candidate modes for the *i*th intra block is denoted as *Final$_i$*. In each case, the proposed method chooses *Budget$_i$* and *Optimal$_i$* to be *Final$_i$*, and update *Extra* for encoding next intra block. Each case is described below:

1) $Optimal_i \le \lfloor Budget_i \rfloor$

If $Optimal_i$ is much smaller than *Budget$_i$*, it means that the computation budget is enough. To meet the budget as close as possible, the following updates vare performed.

$$\begin{cases} Final_i = \lfloor Budget_i \rfloor \\ Extra += Budget_i - Final_i \end{cases} \tag{9}$$

2) $\lfloor Budget_i \rfloor < Optimal_i \le Budget_i$

If *Optimal$_i$* is slightly smaller than *Budget$_i$*, *Optimal$_i$* is prefferred to be chosen as *Final$_i$* because the computation constraint can be met and the svaed computation can be reserved for encoding the following blocks. As a result, the computation buffer control is as follows:

$$\begin{cases} Final_i = \lfloor Optimal_i \rfloor \\ Extra += Budget_i - Final_i \end{cases} \tag{10}$$

3) $Budget_i < Optimal_i \le \lfloor Budget_i \rfloor + 1$

If *Optimal$_i$* is slightly larger than *Budget$_i$*, it is preferred to be *Final$_i$* because the computation does not exceed the budget too much. Therefore, we update *Final$_i$* as in (12)

$$Final_i = \lfloor Optimal_i \rfloor \tag{11}$$

4) $\lfloor Budget_i \rfloor + 1 < Optimal_i$

If *Optimal$_i$* is much larger than *Budget$_i$*, it means that the computation budget is not enough. To meet the budget as close as possible, *Final$_i$* is updated as follows:

$$Final_i = \lceil Budget_i \rceil \tag{12}$$

Case 4 is the worst case in the proposed method, because the available computations are too low to satisfy the optimal number of candidate modes. If there are many extra computations when case 4 is chosen, then it can change to case 3 to fit the optimal value of candidate modes.

$$\begin{cases} Final_i = \lfloor Optimal_i \rfloor, \text{when } Extra > \left(\lfloor Optimal_i \rfloor - \lceil Budget_i \rceil\right) \\ Final_i = \lceil Budget_i \rceil, \text{otherwise} \end{cases} \tag{13}$$

D. Computational Buffer Update

After encoding all 4×4 intra blocks in the *j*th frame, we define *Sum$_j$* which is used to record the total number of candidate modes for RDO computation for the *j*th frame.

$$Sum_j = \sum_{i=0}^{M-1} Final_i , \tag{14}$$

where *M* denotes the total number of 4×4 blocks in the *j*th frame. Subsequently we update the two control parameters *Buffer* and *C* in the computation buffer as in (15) for encoding the next frame.

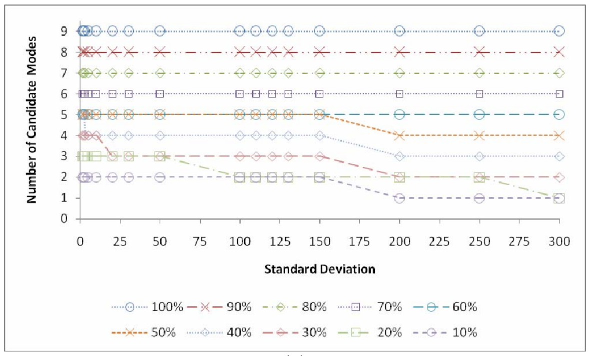
$$\begin{cases} Buffer = Buffer - Sum_j \\ C = C - M \end{cases} \tag{15}$$

The flowchart of the proposed computation control algorithm is depicted in Figure 5.

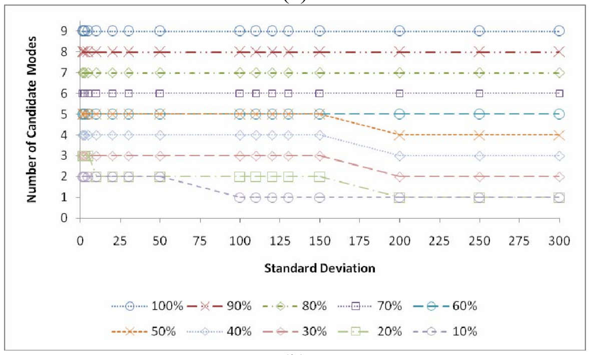
III. Experimental Results

The proposed computation control algorithm is implemented into JVT JM 12.2 reference software, and compared with the original H.264 encoder with full search. In our experiments, the environment settings are as follow:

- Intra Period is set to 1. (All frame using intra coding only)
- Main profile is adopted with RDO and CABAC enabled.
- Each 300-frame sequence is encoded with four quantization parameters: 20, 24, 28, 32, 36, and 40.
- $TH_\sigma = 1$ for each test sequences.



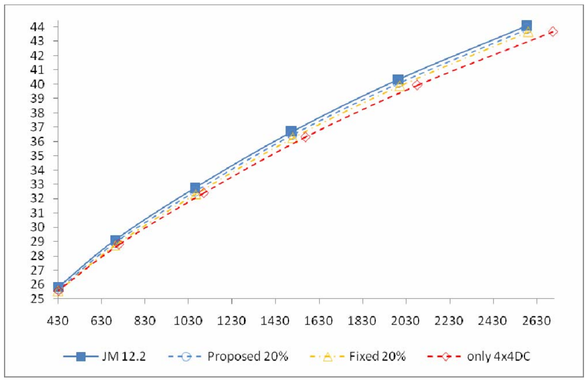
(a)



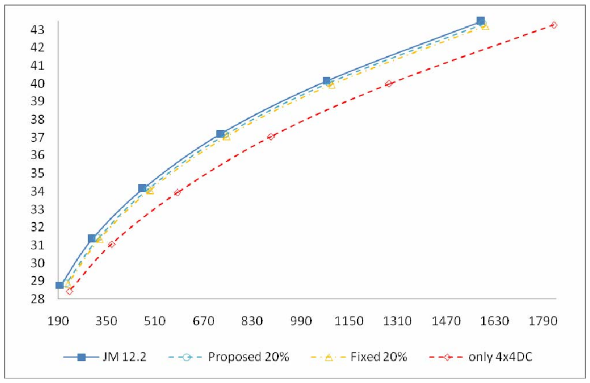
(b)

Figure 6. Average number of candidate modes using the proposed algorithm for QCIF sequences: (a) *Foreman*, QP=40, (c,d)=(5.45, -0.61); (b) *Coastguard*, QP=20, (*c*,*d*)=(3.73, -0.42))

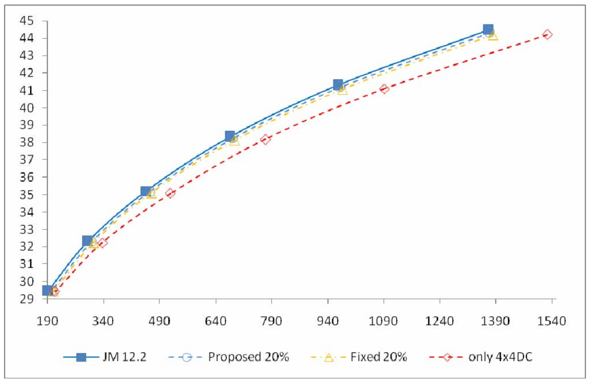
Figure 6 shows the average number of candidate modes by using our proposed method with various constraints. Under each computation constrain, the proposed algorithm always makes a decision that fits the trend of SR curve and assigns the computation resources properly. Table III lists the average time saving with various computation constraints. The intra 4x4 mode decision consumes about 60% of the total encoding time according to the experiments, and our algorithm almost fit this condition with the optimal coding efficiency.



(a)



(b)



(c)

Figure 7. The RD-performance comparison between the original H.264 encoder with proposed computation control algorithm for three QCIF test sequences: (a) *Stefan*, (b) *Foreman*, and (C) *Carphone*.

Figure 7 shows the RD-performance results for three QCIF 300-frame test sequences. We compare the proposed method with three different approaches. The blue square lines indicate the RD-performance of the original H.264 encoder. The light orange triangles represent the method that uses the fixed number of candidate modes in the encoding procedure. The fixed numbers of prediction modes are determined by the constraints. For example, if the computation constraint is 50%, the fixed number of prediction modes is 4.5 on average. The red diamond lines show the modified H.264 encoder that use only the DC mode and disables all other 4x4 intra mode for intra 4x4 blocks. According to the PSNR comparison, we can see that ever though only 20% computation constraint is given, compared with the original H.264 encoder, the PSNR loss of the proposed method is only smaller than 0.24 dB in the worst case. This means that our computational control algorithm can maintain the acceptable video quality when the computational resource is limited.

Table III. Average Time Saving with various computation constrains

Sequences	80%	60%	40%	20%
Foreman	12.55%	25.35%	34.94%	48.65%
Akiyo	11.92%	25.21%	34.02%	47.85%
Carphone	12.16%	25.61%	34.53%	48.58%
Coastguard	13.12%	26.85%	35.62%	50.09%

IV. Conclusions

We proposed a computation-aware intra mode decision for H.264 coding and transcoding applications. The proposed algorithm optimizes the visual quality by adaptively adjusting the number of prediction modes in mode decision under a given computation constraint. We introduced a new concept of computation buffer and formulate the computation control of mode decision as a rate-distortion optimization problem of computation buffer control. Experimental results show that our proposed algorithm can effectively control the computational complexity while maintaining good RD-performance and satisfying the given computation constraint.

REFERENCE

[1] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the H.264/AVC video coding standard,” *IEEE Trans. Circuits Sys. Video Technol.,* vol. 13, no. 7, pp. 560-576, July 2003.

[2] C. S. Kannangara, I. E. G. Richardson, M. Bystrom, J.R. Solera, Y. Zhao, A. MacLennan, and R. Cooney, “Low-complexity Skip Prediction for H.264 Through Lagrangian Cost Estimation,” *IEEE Trans. Circuits Sys. Video Technol.*, vol. 16, no. 2, pp. 202-208, Feb. 2006.

[3] C. S. Kannangara and I. E. G. Richardson, “Computational control of an H.264 encoder through Lagrangian cost function estimation,” in *Int. Workshop Very Low Bit-rate Video-coding*, Sept. 2005.

[4] Y. Wang and S.-F. Chang, “Complexity adaptive H.264 encoding for light weight streams,” *Proc. IEEE Int. Conf. Acoustics, Speech Signal Processing*, May 2006.