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

Jih-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 recent 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.

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

In H.264, if all frames are intra-coded, the main complexity 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 sequence.

2) Buffer: the number of available 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/SATD prediction errors, the SAD/SATD prediction errors of nine prediction modes are sorted by the error values. Figure 1 illustrates the percentages of the rank of the SAD and SATD prediction error (among the nine prediction modes) of the best mode obtained by mode decision with RDO for four test sequences. The figures show that the percentage of the four top-ranking prediction modes (i.e., with smallest SAD or SATD values) being chosen as the best coding mode after mode decision with RDO is about 90%. Such high percentages imply that SAD or SATD prediction error can be used to predict the best mode efficiently without sacrificing the accuracy significantly. In our method, SATD is adopted to calculate the prediction error as it has better accuracy than SAD in predicting the best coding mode.

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

The standard deviation \( \sigma \) of nine prediction modes’ SATD values can be obtained from (1):

\[
\sigma = \sqrt{\frac{1}{9} \sum_{i=0}^{8} (SATD_i - \overline{SATD})^2}, \quad \text{and} \quad \overline{SATD} = \frac{1}{9} \sum_{i=0}^{8} SATD_i
\]  

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’s SATD value. For example, if we find that the average rank of the best mode’s SATD is “4” when 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.

\[
\begin{align*}
\text{Optimal}_{i} &= a + b \cdot \sigma, & \text{for } \sigma = 0 \\
\text{Optimal}_{i} &= c + d \cdot \ln(\sigma), & \text{for } \sigma > 0
\end{align*}
\]

where \( \text{Optimal}_{i} \) denotes the optimal number of candidate modes. The mode parameter sets \((a,b)\) and \((c,d)\) are determined using the least squares method, as shown in (3). Table I lists the derived...
coefficient set \((a,b)\) and \((a,b)\) by using (3) in the Foreman and Carphone sequences.

\[
M = \begin{bmatrix}
1 & \sigma_1 \\
1 & \sigma_2 \\
\vdots & \vdots \\
1 & \sigma_n
\end{bmatrix} \quad \text{and} \quad y = \begin{bmatrix} N_1 \\ N_2 \\ \vdots \\ N_n \end{bmatrix} \quad \text{and} \quad v = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}
\]

\[
\Rightarrow v = (M^T M)^{-1} M^T y
\]

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

<table>
<thead>
<tr>
<th>Sequence</th>
<th>QP</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreman</td>
<td>28</td>
<td>2.3688</td>
<td>0.9272</td>
<td>3.6651</td>
<td>-0.3942</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>2.3493</td>
<td>0.9411</td>
<td>4.2650</td>
<td>-0.4981</td>
</tr>
<tr>
<td></td>
<td>36</td>
<td>2.1755</td>
<td>1.0469</td>
<td>5.0111</td>
<td>-0.6179</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>2.0805</td>
<td>1.0266</td>
<td>5.6999</td>
<td>-0.7094</td>
</tr>
<tr>
<td>Carphone</td>
<td>28</td>
<td>2.1590</td>
<td>0.9037</td>
<td>3.3406</td>
<td>-0.3243</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>2.1271</td>
<td>0.9581</td>
<td>3.3406</td>
<td>-0.3243</td>
</tr>
<tr>
<td></td>
<td>36</td>
<td>2.0500</td>
<td>1.0230</td>
<td>4.1852</td>
<td>-0.4503</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>2.0310</td>
<td>1.0434</td>
<td>5.0347</td>
<td>-0.7094</td>
</tr>
</tbody>
</table>

Figure 2 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.

![Comparison between the actual data and the proposed model](image)

**TABLE II. Coefficient Difference for Foreman QCIF**

<table>
<thead>
<tr>
<th>QP</th>
<th>(CD_1)</th>
<th>(CD_5)</th>
<th>(CD_{All})</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(C)</td>
<td>(D)</td>
<td>(C)</td>
</tr>
<tr>
<td>20</td>
<td>75.28</td>
<td>14.66</td>
<td>57.80</td>
</tr>
<tr>
<td>24</td>
<td>60.53</td>
<td>11.59</td>
<td>49.13</td>
</tr>
<tr>
<td>28</td>
<td>63.79</td>
<td>12.24</td>
<td>53.43</td>
</tr>
<tr>
<td>32</td>
<td>82.21</td>
<td>15.28</td>
<td>67.41</td>
</tr>
<tr>
<td>36</td>
<td>85.42</td>
<td>16.00</td>
<td>62.09</td>
</tr>
<tr>
<td>40</td>
<td>86.98</td>
<td>17.84</td>
<td>63.44</td>
</tr>
</tbody>
</table>

**C. Computation Allocation**

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

\[
Budget_j = \frac{Buffer}{C}.
\]

Before encoding a \(4 \times 4\) intra block, we define two control parameters as below:

1) **Budget**: the computation budget for the \(i\)th \(4 \times 4\) intra block.
2) **Extra**: After encoding \(i\)th \(4 \times 4\) intra block, this control parameter is used to record the surplus computational budget for encoding next block.
While initializing $Budget_i$, 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 3, 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 ($TH$) and there are some extra computations, the $Budget_i$ can be increased. The value of $TH$ is determined empirically.

Finally, the initialization of $Budget_i$ is described in (9).

$$
\begin{align*}
    & Budget_i = Budget_i + 1, \text{ when } \sigma < TH \text{ and } Extra > 1 \\
    & Budget_i = Budget_i - 1, \text{ when } \sigma < TH \text{ and } Extra < 1 \\
    & Extra = Extra, \text{ otherwise}
\end{align*}
$$

(9)

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 $Optimal_i$.

![Flowchart of the Proposed Method](image)

**Figure 4. Flowchart of the Proposed Method.**

![Standard Deviation-Rank Curve](image)

**Figure 3. Standard Deviation-Rank Curve (Foreman, QCIF, 300 I-Frames, QP=40)**
There are four cases of the relationship between $\text{Budget}_i$ and $\text{Optimal}_i$. The final number of candidate modes for the $i$th intra block is denoted as $\text{Final}_i$. In each case, the proposed method chooses $\text{Budget}_i$ and $\text{Optimal}_i$ to be $\text{Final}_i$, and update $\text{Extra}_i$ for encoding next intra block. Each case is described below:

1) $\text{Optimal}_i \leq \text{Budget}_i$

If $\text{Optimal}_i$ is much smaller than $\text{Budget}_i$, it means that the computation budget is enough. To meet the budget as close as possible, the following updates are performed.

$$\text{Final}_i = \lfloor \text{Optimal}_i \rfloor$$

$$\text{Extra}_i = \text{Budget}_i - \text{Final}_i$$

2) $\lfloor \text{Budget}_i \rfloor < \text{Optimal}_i \leq \text{Budget}_i$

If $\text{Optimal}_i$ is slightly smaller than $\text{Budget}_i$, $\text{Optimal}_i$ is preferred to be chosen as $\text{Final}_i$ because the computation constraint can be met and the saved computation can be reserved for encoding the following blocks. As a result, the computation buffer control is as follows:

$$\text{Final}_i = \lfloor \text{Optimal}_i \rfloor$$

$$\text{Extra}_i = \text{Budget}_i - \text{Final}_i$$

3) $\text{Budget}_i < \text{Optimal}_i \leq \lfloor \text{Budget}_i \rfloor + 1$

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

$$\text{Final}_i = \lfloor \text{Optimal}_i \rfloor$$

4) $\lfloor \text{Budget}_i \rfloor + 1 < \text{Optimal}_i$

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

$$\text{Final}_i = \lfloor \text{Budget}_i \rfloor + 1$$

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.

$$\text{Final}_i = \lfloor \text{Optimal}_i \rfloor \text{ when } \text{Extra} > (\lfloor \text{Optimal}_i \rfloor - \lfloor \text{Budget}_i \rfloor)$$

$$\text{Final}_i = \lfloor \text{Budget}_i \rfloor \text{ otherwise}$$

D. Computational Buffer Update

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

$$\text{Sum}_j = \sum_{i=0}^{M-1} \text{Final}_i$$

where $M$ is the total number of 4×4 blocks in the $j$th frame. Subsequently we update the two control parameters $\text{Buffer}$ and $C$ in the computation buffer as in (16) for encoding the next frame.

$$\text{Buffer} = \text{Buffer} - \text{Sum}_j$$

$$C = C - M$$

The flowchart of the proposed computation control algorithm is depicted in Figure 4.

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.
- $TH_0 = 1$ for each test sequences.

Figure 5 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 time of 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.

Figure 6 to Figure 8 show the RD-performance results. 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 even 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

<table>
<thead>
<tr>
<th>Sequences</th>
<th>80%</th>
<th>60%</th>
<th>40%</th>
<th>20%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreman</td>
<td>12.55%</td>
<td>25.35%</td>
<td>34.94%</td>
<td>48.65%</td>
</tr>
<tr>
<td>Akiyo</td>
<td>11.92%</td>
<td>25.21%</td>
<td>34.02%</td>
<td>47.85%</td>
</tr>
<tr>
<td>Carphone</td>
<td>12.16%</td>
<td>25.61%</td>
<td>34.53%</td>
<td>48.58%</td>
</tr>
<tr>
<td>Coastguard</td>
<td>13.12%</td>
<td>26.85%</td>
<td>35.62%</td>
<td>50.09%</td>
</tr>
</tbody>
</table>

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