A FAST CABAC RATE ESTIMATOR FOR H.264/AVC MODE DECISION

Jongmin Hahm, Jaemoon Kim and Chong-Min Kyung

School of Electrical and Computer Engineering & Computer Science
Division of Electrical Engineering
Korea Advanced Institute of Science and Technology

ABSTRACT

H.264/AVC coders use the rate-distortion (R-D) cost function to decide the coding mode for each coding unit for better R-D tradeoff. To evaluate the R-D cost function, both the bit rate and the video quality degradation of the candidate mode must be calculated. In this paper, a fast context-adaptive binary arithmetic coding (CABAC) rate estimation scheme is proposed to accelerate the rate calculation. The speed of the proposed rate estimator depends only on the number of contexts used in the coding unit. Experimental results show that the proposed rate estimator reduces about 20% of the computational complexity of the R-D optimized mode decision when it is implemented as software. The entire encoder implemented as software is then accelerated by 16% with negligible degradation in the R-D performance. If implemented as hardware, the proposed scheme is expected to accelerate the rate estimation for a macroblock by 5 to 18 times faster than the conventional CABAC operation.

***Index Terms*—** H.264/AVC, mode decision, rate-distortion optimization (RDO), estimation, CABAC.

1. INTRODUCTION

Two important operations performed to improve the rate-distortion (R-D) performance of H.264/AVC coders are the mode decision based on the R-D tradeoff and the context-adaptive binary arithmetic coding (CABAC). Each of them saves about 10% of the total bit rate [1, 2]. However, they require a significant additional computation, especially when they are employed together. Evaluating the R-D cost function for each candidate mode is the major source of the additional computational complexity. Several algorithms have been suggested to accelerate the evaluation [3–5], but they are based on context-adaptive variable length coding (CAVLC), not CABAC. One of the recent studies [6] focused on the rate of the CABAC, but its application is limited to intra-mode decisions. A general rate estimation scheme based on CABAC is still needed.

CABAC is a component of an H.264/AVC coder which is often implemented as hardware for fast operation. The speed of CABAC hardware is usually limited by its sequential nature; the context information and internal variables of the arithmetic encoder such as interval range are updated after coding each input symbol. State-of-the-art CABAC coders can process about two symbols per cycle [7] which is still not sufficient for the real time mode decision considering the R-D cost. Computational complexity of the rate calculation was reduced by modeling the behavior of the arithmetic coder [8]. Although the computation time required for obtaining the rate was considerably reduced, this rate estimator still suffers from the same problem as conventional CABAC encoders when implemented as hardware for further speed-up because the context models are still maintained as in the original CABAC algorithm.

In this paper, we propose a rate estimator which approximates the context modeling part of CABAC, based on the rate model of CABAC derived in [8]. The R-D optimized mode decision is reviewed in Section 2. In Section 3, rate estimation methods for consecutive the most probable symbols (MPSs) and the least probable symbols (LPSs) are proposed and they are combined to process sequences with arbitrary length. Experimental results are shown in Section 4 and, finally, conclusions are given in Section 5.

2. R-D OPTIMIZED MODE DECISION

In H.264/AVC reference software, the mode decision is performed with either the low-complexity (LC) cost function or the high-complexity (HC) cost function. The mode decision with the HC cost function is called the R-D optimized mode decision and the cost function is called the R-D cost function. The R-D optimized mode decision selects proper coding options for each coding unit, e.g., macroblock, based on the Lagrange multiplier technique. The R-D cost function is expressed as

$$J = D + \lambda R \tag{1}$$

where λ is the Lagrange multiplier, D is the distortion induced during the encoding process, and R is the number of coded bits. This cost function is evaluated for each candidate coding mode to choose the one with the least. The distortion, D, is measured as the *sum of squared differences* (SSD) between the original coding unit and the reconstructed one. The rate, R, is the length of the coded data after transform, quantization and entropy coding operations. These operations are the most complex part of the R-D optimized mode decision.

3. PROPOSED RATE ESTIMATOR

The input data of CABAC is composed of syntax elements, each of which is coded using certain number of contexts [2]. To begin the estimation, we first group input symbols that use the same context as a data segment. Then the entire input data can be considered as a row of data segments. (see Fig. 1 for an example.) To simplify the rate estimation process, we collect data segments using the same context to form a segment group, and the processing order between data segments ignored. With this modification, we can utilize the correlation between neighboring input symbols in a segment group. This manipulation can be made without loss of accuracy if the rate estimation table of [8] is used. Because variables of the arithmetic coder such as interval range need not be managed, the rate of segment groups are independent of one another.

Because there is no dependency among segment groups, the rate of each segment group can be obtained independently. If we assume

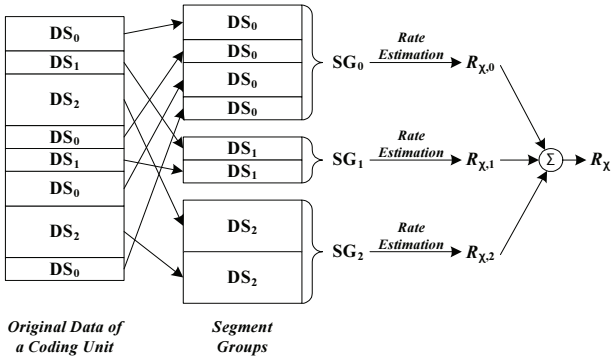

Fig. 1. An illustrative example showing the concept of the proposed rate estimator and related terms. DS_n is a set of consecutive symbols using the n-th context, and SG_m stands for the m-th segment group. It is assumed that the coding unit data in this example uses three contexts for CABAC encoding.

a coding unit χ uses n_χ context models, the rate for the coding unit data can be expressed as

$$R_\chi = \sum_{j=1}^{n_\chi} R_{\chi,j} \tag{2}$$

where $R_{\chi,j}$ is the rate of the segment group using the j-th context model. Our approach is to estimate each value of $R_{\chi,j}$ in a constant time, regardless of the constituent number of bits. Since the number of contexts used for a coding unit is much smaller than the number of symbols, the computational effort required to estimate the rate for a coding unit can be drastically reduced.

3.1. Handling Consecutive MPSs

If there are consecutive MPSs in a segment group, calculating the state index σ after coding the sequence is straightforward because the state index is incremented by one for each MPS input. If we denote the number of consecutive MPSs as n_{MPS}, σ_{new}, the state index after processing n_{MPS} MPSs, is calculated by the formula

$$\sigma_{\mathrm{new}} = \min(\sigma_{\mathrm{old}} + n_{\mathrm{MPS}}, 62) \tag{3}$$

where σ_{old} is the initial state.

For the rate estimate, we look up the estimation table just once for the whole sequence and multiply it by the length of the sequence. The representative table entry is selected with the following equation:

$$\overline{\sigma} = \min(\sigma_{\mathrm{old}} + \lfloor n_{\mathrm{MPS}}/2 \rfloor, 62) \tag{4}$$

The rate estimate for the MPS sequence is now expressed as

$$B(\mathbf{S}_{\mathrm{MPS}}) = \lfloor W(\overline{\sigma}, \mathrm{MPS}) \cdot n_{\mathrm{MPS}}/8 \rfloor \tag{5}$$

where $\mathbf{S}_{\mathrm{MPS}}$ is a bit sequence vector of dimension n_{MPS} which consists of the MPSs only, $B(\mathbf{S}_{\mathrm{MPS}})$ is the estimated number of bits generated by $\mathbf{S}_{\mathrm{MPS}}$, and W is a function corresponding to the rate estimation table (the one defined in [8]) lookup.

3.2. Handling Consecutive LPSs

The method described above cannot be directly extended to a sequence of the LPSs since the state transition is not regular as in the MPS case. To resolve this problem, the updated σ is now approximated, instead of calculating its exact value. First, we divide 63 different σ's into groups as in Fig. 2. The group number for each state represents the number of consecutive LPSs required to bring the state to 0. The group offset is sequentially assigned to each state in a group, such that a larger integer is assigned to a state with larger index. The arrows show the approximated state transition; the transitions are different from the ones defined in the standard, but has a regular structure. Based on this graph, the state after coding a sequence can be estimated by a few arithmetic and logical operations. When there are n consecutive LPSs, the group number is decreased by n and the updated group offset is approximated by n-bit right shifting of the old one. Denoting the group number of σ_{old} by $G_{\mathrm{num}}(\sigma_{\mathrm{old}})$, the group offset by $G_{\mathrm{off}}(\sigma_{\mathrm{old}})$, and the function mapping the group number and offset onto the state by $G^{-1}(\mathrm{number}, \mathrm{offset})$, we can estimate σ_{new} by

$$\sigma_{\mathrm{new}} = G^{-1}(G_{\mathrm{num}}(\sigma_{\mathrm{old}}) - n_{\mathrm{LPS}}, G_{\mathrm{off}}(\sigma_{\mathrm{old}}) \gg n_{\mathrm{LPS}}) \tag{6}$$

provided that the MPS is not changed, i.e., $G_{\mathrm{num}}(\sigma_{\mathrm{old}}) \geq n_{\mathrm{LPS}}$ holds. Otherwise, (6) is substituted by (7) and the MPS change is recorded.

$$\sigma_{\mathrm{new}} = n_{\mathrm{LPS}} - G_{\mathrm{num}}(\sigma_{\mathrm{old}}) \tag{7}$$

These equations provide a method for estimating the state without sequentially processing the state transition for each LPS input.

For the rate estimate, we combine the entries of the rate estimation table of [8] with the probability for each state to obtain the estimated number of bits for each group. Expressing the sum of probabilities for state indices in a certain group $\widehat{G}$ as $P = \sum_{s \in \widehat{G}} p_\sigma(s)$, the rate estimate for $\widehat{G}$ is formulated as

$$B_G(\widehat{G}) = \sum_{s \in \widehat{G}} \frac{p_\sigma(s) \cdot B(s, \mathrm{LPS})}{P}. \tag{8}$$

The resultant rate estimates and corresponding quantized weights W_G are shown in Table 1.

Now that the estimation table is established, we need to determine which entry to read. We first choose a group with the formula;

$$\overline{G} = \max(G_{\mathrm{num}}(\sigma_{\mathrm{old}}) - \lfloor \min(n_{\mathrm{LPS}}, G_{\mathrm{num}}(\sigma_{\mathrm{old}}))/2 \rfloor, 0) \tag{9}$$

Then the number of bits generated by n_{LPS} consecutive LPSs is estimated by

$$B(\mathbf{S}_{\mathrm{LPS}}) = \left\lfloor (W_G(\overline{G}) \cdot n_{\mathrm{LPS}})/8 \right\rfloor. \tag{10}$$

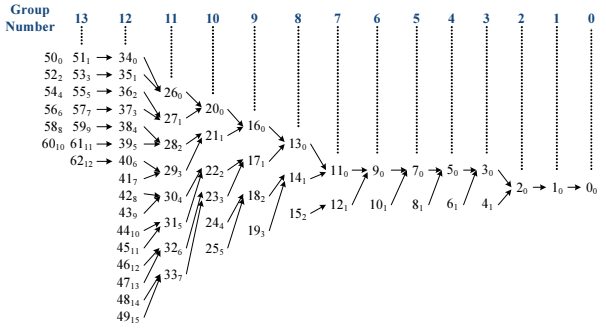

Fig. 2. The approximated state transition diagram for LPS inputs with the group numbers and offsets. The nodes of the diagram represent the state indices σ and their subscripts denote the group offsets. Group numbers are shown at the top.

Table 1. The Rate Estimation Table for LPS Groups

Group $\widehat{G}$	Rate estimate $B_G(\widehat{G})$	Weight $W_G(\widehat{G})$
0	1.00	8
1	1.00	8
2	1.00	8
3	1.34	11
4	1.34	11
5	1.60	13
6	1.71	14
7	1.82	15
8	2.00	16
9	2.34	19
10	2.69	22
11	3.16	25
12	3.95	32
13	5.31	42

However, (10) holds only if the MPS is not changed while processing the sequence. If $G_{\mathrm{num}}(\sigma_{\mathrm{old}}) < n_{\mathrm{LPS}}$ holds, the MPS will be changed and $n_{\mathrm{LPS}} - G_{\mathrm{num}}(\sigma_{\mathrm{old}})$ bits will be processed as the MPSs. Thus, in this case, (10) is substituted by

$$B(\mathbf{S}_{\mathrm{LPS}}) = \lfloor (W_G(\overline{G}) \cdot n_{\mathrm{LPS}})/8 \rfloor + (n_{\mathrm{LPS}} - G_{\mathrm{num}}(\sigma_{\mathrm{old}})). \quad (11)$$

Here we simplified the process for the consecutive MPSs after the MPS change assuming that the weight for each MPS is 8. This assumption causes little error in most cases since the occurrence of the MPS when state index is small often results in one output bit, and the length of consecutive LPSs is usually short such that the number of symbols processed as the MPS is small.

3.3. Handling Arbitrary Input Sequences

Now that we have the estimation methods for both consecutive MPSs and LPSs, we use these algorithms to handle arbitrary input sequences. To this end, we use the number of 0's and 1's (n_0 and n_1, respectively) in the given input sequence. If the context model has 0 (1) as the MPS, n_0 (n_1) is assigned to n_{MPS} and n_1 (n_0) is assigned to n_{LPS}. The exact order of the input symbols is neglected and the sequence is assumed to have a sequence of MPSs followed by a sequence of LPSs. The rate estimate of arbitrary input sequence $\mathbf{S}$ is now expressed as

$$B(\mathbf{S}) = B(\mathbf{S}_{\mathrm{MPS}}) + B(\mathbf{S}_{\mathrm{LPS}}) \quad (12)$$

where B denotes the estimation function and $\mathbf{S}_{\mathrm{MPS}}$ is a bit sequence comprising n_{MPS} MPSs only and $\mathbf{S}_{\mathrm{LPS}}$ is one comprising n_{LPS} LPSs. Since we already have the estimation models for consecutive MPSs and LPSs, $B(\mathbf{S}_{\mathrm{MPS}})$ and $B(\mathbf{S}_{\mathrm{LPS}})$ can be separately calculated. However, the state index, σ, should be updated using (3) before beginning the computation for the LPS part because the MPS part is assumed to have been processed before the LPSs. Using the methods described above, we can obtain the rate estimate for arbitrary sequences.

Despite the ignorance of exact details of the input sequence, this approach yields negligible error. This is because the length of segment group is usually not long enough to yield a significant error. According to a set of experiments conducted on various video sequences, the length of more than 98% of the segment groups was shorter than 16 bits. The additional estimation error caused by this approximation is 0.4 bit on average, which is acceptable. The loss of accuracy when the input sequence is long can be minimized by setting the maximum number of symbols processed at the same time, $l_{\max}$, as a finite number. This parameter was considered to be ∞ until now. For example, if we set $l_{\max} = 16$, a 64-bit-long input sequence is divided into four 16-bit sequences and processed sequentially. With a small $l_{\max}$, the estimation accuracy can be improved while the speed is decreased.

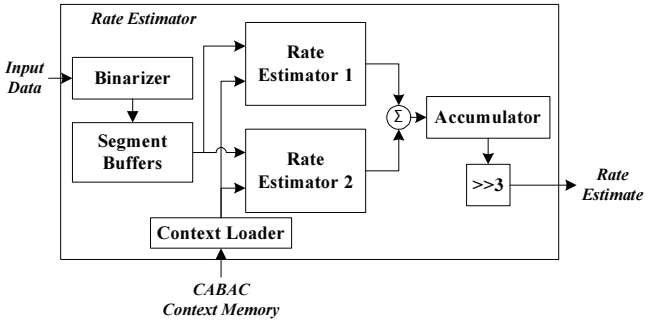

Fig. 3. A block diagram of a hardware architecture using two rate estimators.

After estimating the rate for a coding unit, the resultant state indices can be different from those obtained from the original CABAC algorithm. If such inaccurately updated indices are used as the initial condition for estimating the rate of the next coding unit, the estimation error will be larger for the coding units in the latter part of the slice. On the other hand, after the mode decision of a coding unit, the coding unit undergoes the original CABAC entropy coding, in which the updated state indices are obtained. If the state indices after the entropy coding of the current unit are used to encode the next coding unit, the error propagation can be avoided.

For further improvement in the estimation speed, parallelism can be exploited by using multiple rate estimators at the same time since there is no dependency among segment groups. A hardware architecture using two rate estimators is proposed in Fig. 3. The data for a coding unit is first binarized as in the original CABAC algorithm and organized as segment groups in the segment buffers. The context loader reads contexts for segment groups from the context memory of the CABAC module. The rates independently estimated for each segment group are added up until all segment groups are processed. The rate estimate can be obtained by 3-bit right shifting of the accumulated result. (corresponding to a division by 8 and a floor operation.) If the hardware architecture is carefully designed to achieve a single-cycle throughput per segment group, n_{RE} segment groups can be processed in single cycle with n_{RE} rate estimators. Note that the speed of this hardware is proportional to the number of contexts used in a coding unit, which is usually much smaller than the number of input symbols.

4. EXPERIMENTAL RESULTS

The performance of the proposed rate estimator was evaluated by applying it for the mode decision process of the H.264/AVC reference software. We compared the R-D performance of the mode decision with the proposed rate estimator to the one with the HC cost function defined in JM 10.2. The method of [9] was used with the QPs 28, 32, 36, 40 to obtain ΔPSNR and ΔRate. The time reduction in

Table 2. Experimental Results

Size	Sequence	ΔPSNR (dB)	ΔRate (%)	ΔT_{RD} (%)	$\Delta T_{\mathrm{Total}}$ (%)
QCIF	Carphone	−0.001	0.002	17.17	13.72
	Coastguard	−0.002	0.048	19.29	15.31
	Mobile	−0.001	0.038	26.28	21.62
	Stefan	−0.001	0.002	25.35	20.23
	Football	−0.029	0.919	17.41	13.14
CIF	Bus	−0.004	0.077	20.10	15.80
	Coastguard	−0.005	0.200	17.85	14.05
	Mobile	−0.012	0.335	24.35	20.03
	Stefan	−0.001	0.012	20.48	16.53
	Football	−0.060	1.624	15.32	11.49
Average		−0.010	0.330	20.36	16.19

the R-D cost computation, ΔT_{RD} was computed with the following equation:

$$\Delta T_{\mathrm{RD}} = \frac{T_{\mathrm{originalRDO}} - T_{\mathrm{proposedRDO}}}{T_{\mathrm{originalRDO}} - T_{\mathrm{withoutRDO}}} \times 100\% \qquad (13)$$

where $T_{\mathrm{originalRDO}}$ and $T_{\mathrm{withoutRDO}}$ are the encoding time of the reference code when the RDO option is turned on and off, respectively, and $T_{\mathrm{proposedRDO}}$ is the encoding time when the proposed rate estimation scheme is adopted with the RDO option turned on. The total encoding time reduction is computed by

$$\Delta T_{\mathrm{Total}} = \frac{T_{\mathrm{originalRDO}} - T_{\mathrm{proposedRDO}}}{T_{\mathrm{originalRDO}}} \times 100\%. \qquad (14)$$

One reference frame was used with the motion vector search range of ± 16 and UMHexagonS was adopted for fast ME during the experiments. The GOP structure was IBPBP. Table 2 shows the experimental results from various video sequences. It is shown that usage of the proposed rate estimator can save 20.36% of the R-D cost computation and 16.19% of the total encoding time on average with only 0.33% rate increment or 0.010dB PSNR drop.

The performance gain of the proposed rate estimator hardware is proportional to the ratio of the number of symbols of a coding unit to the number of contexts used. For example, if we assume that the conventional CABAC hardware and the proposed rate estimator have the throughput of 1 symbol/cycle and 1 segment group/cycle, respectively, the speed-up of the proposed rate estimator is approximately expressed by dividing the number of symbols by the number of contexts. It is shown in Table 3 that the maximum number of symbols used in a macroblock is about 5 to 18 times larger than the maximum number of contexts, depending on the test sequence and the QP. Thus, the proposed rate estimator hardware can operate faster than conventional CABAC hardware implementations by a similar ratio, i.e., 5 to 18.

5. CONCLUSION

In this paper, we proposed a rate estimator for the R-D optimized mode decision of H.264/AVC. Experimental data suggested that the proposed estimator can reduce the computational complexity of the R-D cost function by 20.36% and the total encoding time by 16.19% with almost no degradation in R-D performance. The proposed rate estimator can obtain the rate 5 to 18 times faster than conventional CABAC implementations if implemented as hardware. Designing the hardware architecture of the proposed estimator remains as further work.

Table 3. Comparison Between the Maximum Number of Symbols and Contexts

Sequence	QP	Max # Symbols	Max # Contexts	Performance Gain
Carphone (QCIF)	20	1600	129	12.40
	28	912	113	8.07
	36	416	83	5.01
Stefan (QCIF)	20	2313	129	17.93
	28	1289	118	10.92
	36	678	100	6.78
Bus (CIF)	20	1915	131	14.62
	28	1090	119	9.16
	36	550	93	5.91
Stefan (CIF)	20	1838	124	14.82
	28	999	115	8.69
	36	529	90	5.88

6. REFERENCES

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

[2] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 620–636, 2003.

[3] Q. Chen and Y. He, "A fast bits estimation method for rate-distortion optimization in H. 264/AVC," in *Proc. Picture Coding Symp. (PCS 2004)*, San Francisco, CA, 2004, pp. 133–134.

[4] Yu-Kuang Tu, Jar-Ferr Yang, and Ming-Ting Sun, "Efficient rate-distortion estimation for H.264/AVC coders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 5, pp. 600–611, 2006.

[5] M. G. Sarwer and Lai-Man Po, "Fast bit rate estimation for mode decision of H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 10, pp. 1402–1407, 2007.

[6] Shuwei Sun and Shuming Chen, "A novel fast intra-mode decision algorithm for H.264/AVC," in *Image and Signal Processing, 2008. CISP '08. Congress on*, 2008, vol. 1, pp. 324–328.

[7] R. R. Osorio and J. D. Bruguera, "High-throughput architecture for H.264/AVC CABAC compression system," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 11, pp. 1376–1384, 2006.

[8] Jongmin Hahm and Chong-Min Kyung, "Efficient CABAC rate estimation for H.264/AVC mode decision," *IEEE Trans. Circuits Syst. Video Technol.*, submitted for publication.

[9] Gisle Bjontegaard, "Calculation of average PSNR differences between RD-curves," in *Proc. ITU-T Q.6/SG16 VCEG 13th Meeting*, Austin, TX, Apr. 2001, Doc. VCEG-M33.