Binary Arithmetic Coding System with Adaptive Probability Estimation by "Virtual Sliding Window"

Eugeniy Belyaev, Marat Gilmutdinov and Andrey Turlikov

Abstract — *"Virtual Sliding Window" algorithm presented in this paper is an adaptive mechanism for estimating the probability of ones at the output of binary non-stationary sources. It is based on "Imaginary sliding window" idea proposed by B.Ryabko. The proposed algorithm was used as an alternative adaptation mechanism in Context-Based Adaptive Binary Arithmetic Coding (CABAC) - an entropy coding scheme of H.264/AVC standard for video compression. The "virtual sliding window" algorithm was integrated into an open-source codec supporting H.264/AVC standard. Comparison of the "virtual sliding window" algorithm with the original adaptation mechanism from CABAC is presented. Test results for standard video sequences are included. These results indicate that using the proposed algorithm improves rate-distortion performance compared to the original CABAC adaptation mechanism. Besides improvement in rate-distortion performances the "Virtual Sliding Window" algorithm has one more advantage. CABAC uses a finite state machine (FSM) for estimation of the probability of ones at the output of a binary source. Transitions for FSM are defined by a table stored in memory. The disadvantage of CABAC consists in frequent reference to this table (one time for every binary symbol encoding), which is critical for DSP implementation. The "Virtual Sliding Window" algorithm allows to avoid using the table of transitions [1].*

Index Terms — CABAC, entropy coding, context modeling, binary arithmetic coding, imaginary sliding window, universal coding, universal prediction, H.264, MPEG-4 AVC.

I. Introduction

Adaptive arithmetic encoding is included in most modern standards of video compression. Compression rate of such encoding technique largely depends on precision of encoded symbols probabilities estimation. In the set of algorithms of adaptive probability estimation a class of methods can be distinguished that are used for encoding of stationary sources on limited time intervals. One of the first algorithms of this class is the "sliding window" algorithm. But this algorithm is memory-demanding and not suitable for practical usage in video compression applications. A number of new methods based on "sliding window" idea approximating its work were suggested. Rivest and Leighton [2] and independently Ryabko [5] proposed an algorithm of probability estimation using a randomized finite-state machine. This algorithm is known as "Imaginary Sliding Window" (ISW). Feder and Meron in paper [4] gave an extensive review of estimation techniques using finite memory and suggested a method which consists of randomized finite-state machine replacement by time invariant deterministic finite-state machine (TIDFS). In this paper we suggest a way to implement derandomized method, which we shall call "virtual sliding window". The proposed method is easy in implementation and does not require extra memory storage. This paper gives results of practical application of a "virtual sliding window" in Context-Based Adaptive Binary Arithmetic Coding, which is used in H.264/AVC video compression standard. We also show that the proposed algorithm is efficient in comparison with a TIFDS machine-based technique applied in the standard.

[1] The authors are with the department of information systems, State University of Aerospace Instrumentation, Saint-Petersburg, Russia (e-mail: e_beliaev@mail.ru; gmrt@list.ru; turlikov@mail.ru).

II. Data compression using "sliding window"

Algorithms of adaptive data encoding based on "sliding window" are widely known. The probability of source symbol is estimated by analysis of special buffer contents. It keeps W previous encoded symbols, where W is length of buffer. After encoding of each symbol the buffer contents is shifted by one position, new symbol is written to the free cell and the earliest symbol in buffer is erased. This buffer is called "sliding window" after the method of buffer content manipulation.

For binary sources probability of ones is estimated by Krichevsky-Trofimov [1] formula

$$\hat{p}_t = \frac{S_t + 0.5}{W + 1} \qquad (1)$$

where S_t is number of ones in window at time t.

The advantage of using the "sliding window" is opportunity of precise evaluation of source statistics and fast adaptation to changing statistics. However, the window has to be stored in encoder and decoder memory, which is a serious disadvantage of this algorithm.

1-4244-0216-6/06/$20.00 ©2006 IEEE

There exist algorithms matching "sliding window" performance. These algorithms make use of the fact that the window is necessary only for recount of symbols and it doesn't have to be stored in memory. (As far as the symbol erased from window is unknown at every step, the number of symbols is recounted in accordance with a specific rule). We shall give an overview of such algorithms described in [2], [4] and [5].

First we shall consider "imaginary sliding window" technique (ISW) proposed for a binary source in paper [2] and for non-binary source in Ryabko's paper [5]. The ISW technique does not require window content storage and estimates count of symbols from source alphabet stored in window.

We shall describe the ISW method for a binary source. Define x_t – source output symbol at time instant t, $x_t \in \{0,1\}$; y_t – symbol deleted from the window at time instant t, $y_t \in \{0,1\}$;

Suppose at every time instant a symbol in a random position is erased from the window instead of the last one. Then the number of ones in the window is recalculated by the following recurrent randomized procedure.

Step 1. Delete a random symbol from the window

$$S_{t+1} = S_t - y_t$$

where y_t is a random value generated with probabilities

$$\begin{cases} \Pr\{y_t = 1\} = \dfrac{S_t}{W} \\ \Pr\{y_t = 0\} = 1 - \dfrac{S_t}{W} \end{cases}.$$

Step 2. Add a new symbol from the source

$$S_{t+1} = S_{t+1} + x_t .$$

For implementation of ISW algorithm a random variable must be generated. This random variable should take the same values at corresponding steps of encoder and decoder. However, there is a way to avoid generating a random variable. At step 1 of the algorithm let us replace random value y_t with its probabilistic average. Then the rule for recalculating number of ones after encoding of each symbol x_t can be presented in two steps.

Step 1. Delete an average number of ones from the window

$$S_{t+1} = S_t - \frac{S_t}{W}.$$

Step 2. Add a new symbol from the source

$$S_{t+1} = S_{t+1} + x_t .$$

The final rule for recalculating number of ones can be given as follows

$$S_{t+1} = S_t - \frac{S_t}{W} + x_t = \left(1 - \frac{1}{W}\right) S_t + x_t . \qquad (2)$$

The rule for recalculating (2) was derived in [4] and belongs to a class of algorithms using *time invariant deterministic finite-state* (TIDFS) machine.

III. DESCRIPTION OF "VIRTUAL SLIDING WINDOW" ALGORITHM

The main drawback of the proposed in [4] algorithm is using floating-point operations because S_t is not integer. For transition to an integer algorithm we shall give an interpretation of ISW algorithm as follows. Suppose a "sliding window" of W cells is given. On input of next symbol one cell is chosen at random. The symbol in the chosen cell is replaced by the newly received symbol.

The rule of recalculation (2) in integer implementation can have the following interpretation. Let there be a "sliding window" of cW cells, where c is the algorithm parameter. The value of received symbol is put into c randomly chosen cells and the average number of ones in selected c cells is removed from the window.

Now suppose that the number of ones in the window of cW cells after encoding of next symbol x_t is recalculated by the rule

$$\begin{aligned} S_{t+1} &= S_t + \left\lfloor \frac{cW - S_t + \frac{W}{2}}{W} \right\rfloor , \text{ if } x_t = 1. \\ S_{t+1} &= S_t - \left\lfloor \frac{S_t + \frac{W}{2}}{W} \right\rfloor , \text{ if } x_t = 0. \end{aligned} \qquad (3)$$

Further we shall call recalculation rule (3) a "virtual sliding window" (VSW). Probability estimation of ones for binary source output for VSW algorithm implementation is defined as

$$\begin{aligned} \hat{p}_t &= \frac{S_t}{cW} , \text{ if } x_t = 1. \\ \hat{p}_t &= 1 - \frac{S_t}{cW}, \text{ if } x_t = 0. \end{aligned} \qquad (4)$$

Implementation of VSW algorithm permits to avoid:

- storing the "sliding window" in encoder and decoder memory;
- random value generation;
- floating-point operations.

If $W = 2^i$ is chosen where i is a positive integer, division can be replaced by shift operation.

As a criterion for choice of parameter c we shall take the equality of minimal probability estimation of ones for binary source output for ISW and VSW algorithms. From (1) it follows that the minimal estimated probability for ISW algorithm is

$$\hat{p}_{\min} = \frac{1}{2(W+1)}. \qquad (5)$$

From (3) and (4) it follows that minimal estimated probability for VSW algorithm is

$$\hat{p}_{\min} = \frac{\frac{W}{2} - 1}{cW}. \quad (6)$$

Equating (5) and (6) gives

$$\frac{1}{2(W+1)} = \frac{\frac{W}{2} - 1}{cW}.$$

$$c = W - 1 - \frac{2}{W} \approx W \ \ (\text{if} \quad W >> 1).$$

Using formula (4) is possible after W symbols entered the window. Therefore, in initial time interval $0 < t < W$ the probability estimation is calculated by Krichevsky-Trofimov formula

$$\hat{p}_t = \frac{n_t + 0.5}{t+1} \quad (7)$$

where n_t is the number of ones received in time interval $[0, t]$; next, at time moment $t = W - 1$ initialization of ones number in VSW algorithm is carried out

$$S_W = \hat{p}_{W-1} cW = \frac{(n_{W-1} + 0.5)}{W} cW = Wn_{W-1} + \frac{W}{2}$$

where $\hat{p}_{W-1}$ is the probability estimation obtained by formula (7) at time $t = W - 1$. At time instants $t \geq W$ probability estimation is calculated by formula (4).

IV. Using CABAC in H.264/AVC standard

In standard H.264/AVC two types of entropy encoding are used: Context-based Adaptive Binary Arithmetic Coding (CABAC) and Context-based Adaptive Variable Length Coding (CAVLC). In paper [3] it is shown that the bit-rate savings of CABAC can be as high as 20%. But encoding time for CAVLC is significantly less than that of CABAC. Therefore, using CABAC in time-critical applications (e.g. real-time applications) is problematic. As it will be shown later, "virtual sliding window" (VSW) is one of the ways to improve CABAC performance. With VSW it is also possible to improve compression in comparison with the probabilities estimation method used in current version of CABAC.

The CABAC algorithm consists of three successively run steps.

- Binarization
- Selection of context model for encoded binary symbol
- Encoding by an arithmetic coder and adaptation of the context model used at the previous step.

Binarization is mapping a non-binary symbol from CABAC input into a binary sequence called a *binary string*. In H.264/AVC several binarization types are used, but the primary method after which binary string is encoded in regular mode is unary binarization. Unary binarization maps a non-negative value x to a sequence of x zeros and a terminating one.

For each symbol from binary string generated at previous step a context model is selected from a fixed set in accordance with a specific rule. Each context model contains a description on a binary (Bernoulli) source (in fact, a current estimation of distribution of this source). The selection points to source that the output currently encoded symbol belongs to.

Encoding of a binary symbol can be carried out in one of two modes:

-Regular, when a symbol is encoded with probability estimations specified by the selected context model. After encoding, probability estimations of this context model are updated.

-Bypass, when a model with fixed probability estimations of 0.5 for both zeros and ones is used. In this mode no probability estimation are updated.

The algorithm of estimates adaptation for regular mode is based on TIDFS machine. CABAC has 128 probability states and transition table with two parts: one - for symbol with the lowest probability (Least Probable Symbol – LPS) and second for symbol with the most probability (Most Probable Symbol – MPS) at the current moment. As far as a FSM of the same type is used for each context model, the transition table of this FSM is stored in memory only once. However, the current state of this machine is to be stored for each context model. When probability estimation for the present context model is modified, transition table lookup is required in order to define the next state and exchange LPS and MPS definitions if necessary.

Let us mark some features of adaptation procedure implementation used in CABAC and entropy encoding in H.264/AVC on the whole.

- Modification of FSM states and consequently of probability estimations for all context models is carried out in accordance with the same rules. In other words, the present algorithm does not take account of specific features of different context models.
- Each context model has its own initial state of the FSM for initialization at the start of encoding. Initial values have to be stored either in file or in memory.
- At the start of each frame processing, initialization of FSM states is performed for all context models regardless of the frame type.

V. Features of VSW algorithm implementation in H.264/AVC standard

In common case the binary sources corresponding to these models are non-stationary. Statistical properties of these sources can vary a great deal. This fact makes the problem of choosing window length for each source very significant.

For stationary memoryless sources window length expansion increases probability estimation precision and improves compression rate. For arbitrary source window length expansion may reduce estimation precision.

Optimal window length selection is a complex problem because statistical properties of sources corresponding to context models are unknown *a priori*. The simple heuristic algorithm of window length selection is proposed. Let us define a $L=\{w_1, w_2, ..., w_l\}$ of window length. The test sequence is encoded and then we select window length for each context model from the set L. Let us consider context model with number i. During the test sequence encoding we calculate probability estimations $\hat{p}_j(i,w_1),...,\hat{p}_j(i,w_l)$ for symbol with number j corresponding to this model. After test sequence encoding, bitstream length estimation is calculated by equation

$$R(i,w_k) = -\sum_j \log \hat{p}_j(i,w_k) \ .$$

Window length $W(i)$ for context model with number i is assigned by equation

$$W(i) = \min_k R(i,w_k) \ .$$

Sometimes it makes sense to use the same window length W for all context models. Then window length selection depends only on the quantization step value. In this case W is chosen subject to minimal total bitstream length estimation calculated among all context models

$$W = \min_k \left(-\sum_i \sum_j \log \hat{p}_j(i,w_k) \right).$$

VI. Practical results of applying VSW algorithm in H.264/AVC standard

To obtain practical results the suggested algorithm VSW was embedded into JVT codec, version JM. 10.2 (FREXt), supporting H.264/AVC standard. As results we show bit-rate savings provided by VSW usage relative to the original version of the codec. PSNR value is the same in both cases.

Modification of the initial algorithms affects two aspects. First, we suggest using two variants of VSW algorithm. The first variant uses the same window length for all context models, while the second chooses a specific window length for each context model. The results are given in tables 1 and 2. Second, we suggest avoiding initialization of context model states on each frame for type P (predicted). Initialization of P-frames is carried out only after encoding of key frames (type I - intra). The latter way of initialization we shall call non-regular. The results of such modification are given in tables 3 and 4. Table 3 shows results obtained using same window lengths for all context models; table 4 shows results for different window lengths. "Foreman" video sequence was used as a test sequence for window length selection. Values of parameter W in both cases were selected from set $L=\{2^3, 2^4, 2^5, 2^6, 2^7, 2^8, 2^9\}$.

Comparing the proposed "virtual sliding window" algorithm with the original CABAC adaptation mechanism for JVT/H.264, average bit-rate savings about 0.1-5.5% can be obtained.

TABLE I
Bit-rate savings providing by usage of VSW method with fixed window length

QP	10	20	30	40	50
foreman	0.58	0.43	0.16	0.10	0.60
mobile	0.32	0.35	0.30	0.06	0.35
tempete	0.16	0.34	0.33	-0.08	0.78

TABLE II
Bit-rate savings providing by usage of VSW method with variable window length

QP	10	20	30	40	50
foreman	0.76	0.73	0.61	0.79	2.35
mobile	0.12	0.28	0.19	0.30	0.64
tempete	0.16	0.40	0.30	0.35	2.06

TABLE III
Bit-rate savings providing by usage of VSW method with fixed window length (non-regular initialization)

QP	10	20	30	40	50
foreman	0.64	0.48	0.50	0.89	4.20
mobile	0.35	0.36	0.47	0.68	2.77
tempete	0.23	0.34	0.36	0.36	4.12

TABLE IV
Bit-rate savings providing by usage of VSW method with variable window length (non-regular initialization)

QP	10	20	30	40	50
foreman	0.81	0.82	1.03	1.74	6.00
mobile	0.19	0.34	0.54	1.17	3.61
tempete	0.20	0.41	0.46	0.85	5.57

VII. Conclusion

The proposed "virtual sliding window" method for probability of ones estimation is a development of ISW method. Like ISW, VSW method approximates work of "sliding window" algorithm but does not require to store the window in memory. An additional advantage of VSW is avoiding random value generation for state modifications. Methods similar to VSW were proposed in other works, e.g. [4], but distinguishing feature of VSW is integer implementation.

As compared to probability estimation technique used in CABAC, the suggested method allows to obtain better compression rate. The compression gain is reached by assigning to different context models specific window lengths selected by statistical properties of corresponding source. Implementation of VSW method is not more complex than CABAC implementation. Moreover, VSW implementation on DSP has additional advantage. It does not need to store transition table for FSM used for modification of probability estimation.

One of the ways for improving VSW method can be usage of adaptive window length during sequence encoding.

REFERENCES

[1] E. Krichevski and V. E. Trofimov, "The performance of universal encoding," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 199–207, Mar.1981.

[2] T. Leighton and R. L. Rivest, "Estimating a probability using finite memory" *IEEE Trans. Inform. Theory*, vol. IT-32, pp. 733–742, Nov.1986.

[3] D.Marpe, H.Schwarz and T.Wiegand, "Context-based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard" *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, No. 7, pp. 620-636, July 2003.

[4] E. Meron and M. Feder, "Finite-Memory Universal Prediction of Individual Sequences" *IEEE Trans. Inform. Theory*, vol. 50-7, pp. 1506–1523, July 2004.

[5] B.Y. Ryabko, "Imaginary sliding window as a tool for data compression," *Probl. Inform. Transm*., pp. 156–163, Jan. 1996.

Eugeniy Belyaev graduated from St.-Petersburg State University of Aerospace Instrumentation (SUAI) in 2005. Since 2005 he has been PhD student of SUAI. His research interests include video compression algorithms and digital signal processing.

Marat Gilmutdinov is an assistant professor of information systems department of St.-Petersburg State University of Aerospace Instrumentation (SUAI). He received his PhD degree at SUAI in 2002. His research interest include still image and video compression algorithms, methods of real-time data transmission over IP networks and algorithms of random multiple access.

Andrey Turlikov is an associate professor of information systems department of St.-Petersburg State University of Aerospace Instrumentation (SUAI). He graduated from Leningrad State Institute of Aerospace Instrumentation (now SUAI) in 1980 and received his PhD degree in 1986. Since 1987 he has been involved in teaching work. Dr. Turlikov is the author of approximately 50 publications. His research interests include multiple access systems, real-time data transmission protocols and video compression algorithms.