Journal of Information Hiding and Multimedia Signal Processing ©2017 ISSN 2073-4212
Ubiquitous International Volume **8**, Number **5**, September **2017**

An Efficiently Adaptive Probability Estimation Method in CABAC of HEVC Using Stochastic Learning Weak Estimator

Hao Chen[1], Tengye Su[2], Ye Zhang[3]

[1]Department of Information Engineering
School of Electronics and Information Engineering
Harbin Institute of Technology, China
Corresponding Author: hit_hao@hit.edu.cn

[2]Department of Electronic Information Engineering
School of Electronics and Information Engineering
Harbin Engineering University, China

[3]Department of Electrical and Communication Engineering
Harbin Institute of Technology, China

Received October, 2016; revised May, 2017

Abstract. *In the coding process of Context-based Adaptive Binary Arithmetic Coding (CABAC) in High Efficiency Video Coding (HEVC), the probability estimation with table lookup is adopted to improve the speed of the estimation at the cost of the compression efficiency. In order to improve the compression efficiency while maintaining the speed, an efficiently adaptive probability estimation method based on the Stochastic Learning Weak Estimator (SLWE) is proposed. Due to the strong adaptability of SLWE to non-stationary data, a basic framework of probability estimation with SLWE is firstly established by multiplication principle on the learning factor instead of look-up table operation. To further enhance the compression efficiency, a new learning factor with the expression of 1-1/2w is designed benefiting from its adaptability to data characteristic and the form of power of 2. Thus, the next estimation probability can be computed by simple operations in place of multiplication, including binary shift of the current symbol's probability and addition or subtraction for the values resulting from the shift. Experimental results on synthetic data and the standard test videos show that in comparison with CABAC, the average PSNR is increased by 0.417 dB or the average bit rate is reduced by 2.11%.*

Keywords: HEVC, CABAC, Probability estimation, SLWE, Learning factor.

1. Introduction.

ADAPTIVE binary arithmetic encoding is used in most modern standards of video and image compression, such as H.264/AVC [1], HEVC [2], JPEG [3], JPEG2000 [4], and Dirac [5]. The encoding performance of arithmetic encoding mainly depends on the probability estimation model on the characteristics of actual sources. In the case of CABAC, the probability estimation is required to constantly judge and to look up tables to obtain the corresponding probability and interval data. As the data in the table is the experience value before performing the experiment, when the tested video sequence is changed, its probability estimation is likely to appear an error. Therefore, it is necessary to improve the quality of compressed data as much as possible while guaranteeing the speed of probability estimation.

In order to estimate the probability of the symbols to be encoded, many approaches have been developed. For stationary data, traditional methods such as maximum likelihood

[6] and Bayesian parameter estimation [7] are often used. The process of maximum likelihood method is very simple. After encoding a symbol for once, the number of the occurrence frequency of the corresponding symbol category is increased by one, and then the category occurrence frequency and the total number of the encoded symbols are used to estimate the probability. In the encoded symbols, the kinds of symbol which have the large frequency, will acquire high occurrence probability in the next process. However, this method is lack of sufficiency theoretical basis, tending to be reasonable by experiences. Besides, Bayesian parameter estimation has a better theory basis, which assumes that the estimated parameters have a prior distribution and it strengthens the possibility of a possible distribution according to continuously observed values. Since the real distribution of the parameters for stationary data is invariable with time, the Bayesian estimation will converge to the real distribution with the probability of 100% with the increase of the amount of the observed data. Namely, for stationary data, the classic Bayesian parameter estimation method is effective.

Moreover, for non-stationary data, some methods including forgetting factor [8] and sliding window [9] are applied. Sliding window method analyzes a specific buffer content to estimate the probability distribution of the source for the current symbol to be encoded. The buffer stores the encoded symbols and its number is W. When encoding a new symbol, the contents of the buffer conducts a shift and then the new symbol is put into the buffer. The symbol which enters the buffer first is deleted. Since most of the actual sources are non-stationary, probability estimation of non-stationary data has received lots of research attentions. To ensure the estimation precision, probability estimation needs to be easily realized with low latency as input symbol's characteristic varies.

In addition, low memory is also necessary in the process of probability estimation. Sliding window [10] is a typical algorithm for encoding of non-stationary sources over limited time intervals, whereas it demands much memory for the window and is not suitable for practical application in video compression applications [11]. Some new methods based on sliding window were suggested. Rivest, Leighton [12] and Ryabko [13] proposed an algorithm of probability estimation using a randomized finite-state machine. It is known as Imaginary Sliding Window (ISW). Feder and Meron [14] gave an extensive review of estimation techniques using finite memory and suggested a method consisting of randomized finite-state machine replacement by time invariant deterministic finite-state machine (TIDFS). Among some new methods derived from sliding window, virtual sliding window (VSW) can estimate the probability using the number of symbols without the order of the symbols in the window. Although VSW doesn't require look-up tables, it still needs multiplication operation. Feder and Meron presented a method using randomized finite-state machine replacement by time invariant deterministic finite-state machine.

In recent years, Learning Automata (LA) theory based weak estimation methods were applied in the probability estimation for non-stationary data. One of the representatives of the parameter estimation methods is Stochastic Learning Weak Estimator (SLWE). Oommen and his collaborators [15] estimated the binomial and polynomial distributions in non-stationary environments. In probability estimation [16], not only SLWE can save the memory of window data cache in those window-based methods, it also has stronger adaptability to the change of non-stationary data and is suitable for various types of data [17].

Motivated by the SLWE technology, an efficiency adaptive probability estimation method is proposed in CABAC of HEVC. We firstly set up a basic framework of probability estimation for non-stationary data using multiplication principle on learning factor in SLWE to replace the judgment and look-up operations in the original CABAC. In order to further improve the estimation precision, the new expression with the power exponent of 2 is adopted as a substitute for learning factor, in this way an adaptive probability estimate is proposed. Accordingly, probability update can be implemented by binary shifting of the probability for the current symbol and addition or subtraction for the values resulting from the shift. Thus, no multiplication operation is required in the process of probability estimation and the storage space would be consequently reduced. This process can be applied to the coding phase of the video compression standard HEVC, and under the condition of keeping the speed, it can effectively improve the coding accuracy.

The rest of this paper is organized as follows. Section II reviews the probability estimation in CABAC of the HEVC standard and SLWE-based probability estimation, and conducts a comparative analysis of principle between SLWE and ISW. Section III introduces the proposed high precision adaptive probability estimation including basic framework of the SLWE-based probability estimation in CABAC of HEVC and its improvement using variable and appropriate learning factor. Section IV evaluates the performance of the proposed method on synthetic data and standard test video sequences, using these main features including coding redundancy, computation complexity, precision of probability estimation, speed of coding computation, and bit-rate savings. Experimental results show that the proposed algorithm achieves a good rate control and produces an improvement in rate distortion. Conclusions are drawn in Section V.

2. Basic description of Probability Estimation In CABAC and SLWE-BASED Method.

2.1. The Probability Estimation in CABAC of HEVC.

Let us consider a stationary discrete memoryless binary source with p denoting the probability of ones. For a binary sequence$x^N = \{x_1, x_2, ..., x_N\}$ (where N is the length of this sequence), $x_t \in \{0, 1\}$ is represented as $\left[-\log_2 P\left(x^N\right) + 1\right]$ bits of a number$Q\left(x^N\right) + P\left(x^N\right)/2$, where '$t$' denotes the current time, $P\left(x^N\right)$ and $Q\left(x^N\right)$ are the probability and the cumulative probability of the binary sequencex^N, respectively x^N can be obtained by the recurrent relations as follows [18]:

$$\begin{cases} Q\left(x^t\right) = Q\left(x^{t-1}\right) \\ P\left(x^t\right) = P\left(x^{t-1}\right)(1-p) \end{cases} x_t = 0 \tag{1}$$

$$\begin{cases} Q\left(x^t\right) = Q\left(x^{t-1}\right) + P\left(x^{t-1}\right)(1-p) \\ P\left(x^t\right) = P\left(x^{t-1}\right)p \end{cases} x_t = 1 \tag{2}$$

An integer implementation of the arithmetic encoder in CABAC of HEVC is based on two registers, i.e., L and R (see Fig. 1, where T is the range occupied by the probability p). The register L corresponds to $Q\left(x^N\right)$ and the register R corresponds to $P\left(x^N\right)$. The precision of the registers L and R grows with the increase of the length of this sequence. In order to decrease the coding latency and to avoid the registers underflow, the renormalization procedure [19] is used for the each output symbol. The original CABAC utilizes the look-up table operation to perform probability estimation. Each time of updating probability needs to read data from tables for twice. When the symbols resulted from quantization process flow into CABAC, the table "TComCABACTables::sm_aucLPSTable[64][4]" is looked up to determine the initial interval and the initial probability. On the basis of symbol belonging to MPS or LPS, the table "ContextModel::m_aucNextStateMPS[64]" or "ContextModel::m_aucNextStateLPS[64]" is used to update MPS or LPS. If there are several symbols to be encoded and the corresponding probabilities to be updated, the tables are looked up for many times, and this leads to a long-time process of probability estimation.

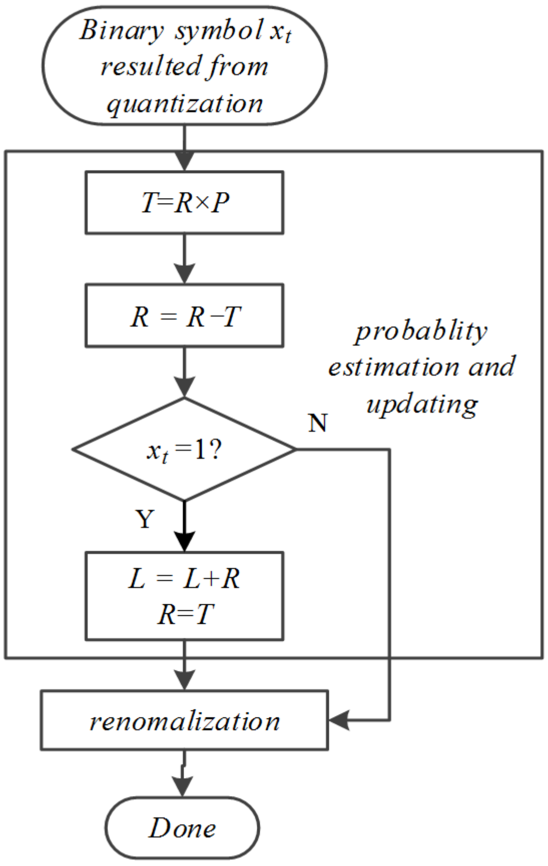
Binary symbol x_t resulted from quantization
T=R×P
R = R−T
x_t=1?
N
Y
L = L+R
R=T
probablity estimation and updating
renomalization
Done

FIGURE 1. x_t encoding procedure

2.2. **SLWE-based Probability Estimation Method.** Suppose that $x_t \in \{0,1\}$ is a specific value of a binary sequence $x^N = \{x_1, x_2, ..., x_N\}$ at time "t". Furthermore, x^N obeys Bernoulli distribution S. In order to estimate S, i.e., s_i for i =0, 1, we maintain a running estimate $P(t) = [p_0(t), p_1(t)]^T$ of S, where $p_i(t)$ is the estimate of s_i at time "t", for i = 0, 1. Then, the value of $p_1(t+1)$ is updated by:

$$p_1(t+1) = \begin{cases} \lambda p_1(t) + (1-\lambda) \text{ if } x_t{=}1 \\ \lambda p_1(t) \text{ if} x_t{=}0 \end{cases} \tag{3}$$

where$\lambda$is a user-defined parameter, 0 ¡ λ ¡ 1. There are several theorems about this process. The first theorem and its proof can be found in [16], which concerns the distribution of the vector $P(t)$ which estimates S as shown in equation (3). Let us state that$P(t)$ converges in distribution. The mean of $P(t)$ is shown to converge exactly to the mean of S. The proof, which is a little involved, follows the types of proofs used in the area of stochastic learning [17].

Theorem 2.1. *Let X be a binomially distributed random variable, and $P(t)$ be the estimate of S at time 't', thus$E[P(\infty)] = S$.*

Theorem 2.2. *Let X be a binomially distributed random variable governed by the distribution S. If $P(t+1)$ is the estimate of S at time 't+1', $E[P(t)]$ obeys an ergodic Markov chain whose steady state distribution converges to S.*

Theorem 2.3. *Let X be a binomially distributed random variable governed by the distribution S, and $P(t)$ be the estimate of S at time 't' obtained by (3). Then, the rate of convergence of $P(t)$ to S is fully determined by λ.*

We now derive the explicit expression for the asymptotic variance of the SLWE. Small value of λ leads to fast convergence and large variance, and vice versa.

Theorem 2.4. *Let X be a binomially distributed random variable governed by the distribution S, and $P(t)$ be the estimate of S at time 't' obtained by (3). Then, the algebraic expression for the variance of $P(\infty)$ is fully determined by λ.*

2.3. **Comparative Analysis of Principle between SLWE and ISW.** Actually, SLWE algorithm has a close relationship with the sliding window method. For sliding window method, a buffer keeps W previously-encoded symbols, where W is the length of the buffer. After encoding each symbol, the buffer's content is shifted by one position. A new symbol is then written into the free cell and the earliest symbol in the buffer is erased [18]. For the binary sources, the probability of ones $\hat{p}_t$ is estimated by the Krichevsky-Trofimov formula [20]

$$\hat{p}_{t+1} = \frac{s_t + 0.5}{W + 1} \tag{4}$$

where s_t is the number of ones in the window before encoding the symbol with the index t. Let us assume that there is a binary source for Imaginary Sliding Window (ISW) method. Define $x_t \in \{0, 1\}$ as the source input symbol with index t, and $y_t \in \{0, 1\}$ as a symbol deleted from the window after adding x_t. Suppose that each time instant a symbol in a random position is erased from the window in the ISW method instead of the last one from the window. The following probability estimation rule of ISW can be obtained:

$$p_{t+1} = \left(1 - \frac{1}{W}\right) p_t + \frac{1}{W} x_t \tag{5}$$

If we assume that$\lambda = 1 - \frac{1}{W}$, equation (6) can be translated into:

$$p_1(t+1) = \lambda p_1(t) + (1-\lambda)\ \ x_t\ x_t \in \{0, 1\} \tag{6}$$

It can be seen that equation (7) is equivalent to the equation (3). Though derivation thoughts of sliding window and SLWE are different. However, the expressions to perform the probability estimation have the same forms in mathematics.

3. **Implementation Of The Fast Adaptive Probability Estimation.** SLWE-based probability estimation doesn't need to look-up table in the process of the probability estimation for CABAC, which will maintain the time of the arithmetic coding. But it still needs multiplication operation, which leads to the decrease of the coding accuracy and also occupies lots of computing time. Therefore, we use the binary operation to replace the multiplication operation and then propose an adjustment method (i.e., update principle) of. Specifically, we provide the implementation of binary arithmetic coder based on SLWE probability estimation method in part A and the improved SLWE probability estimation method based on adaptive in part B.

3.1. **Basic framework of SLWE-based Probability Estimation in CABAC of HEVC.** For state machine based probability estimation in the M-coder [21] of CABAC in the HEVC standard, the input symbols are grouped into two types, i.e., Most Probable Symbols (MPS) and Least Probable Symbols (LPS). The state machine contains 64 states. Each state s defines the probability estimation for LPS. Accordingly, probability values

$$\{\hat{p}_0, \hat{p}_1, ..., \hat{p}_{63}\}$$

are computed by:

$$\begin{cases} \hat{p}_s = \alpha \hat{p}_{s-1} \\ \alpha = \left(\frac{\hat{p}_{\min}}{0.5}\right)^{\frac{1}{63}} \end{cases} \tag{7}$$

where s = 1,...,63, $\hat{p}_0 = 0.5$ and $\hat{p}_{\min} =\ 0.01875$

Probability estimation for symbol is defined as:

$$\hat{p}_{t+1} = \begin{cases} \alpha \hat{p}_t + (1-\alpha)\,, \text{ if } x_t = \text{LPS} \\ \max\{\alpha \hat{p}_t,\ \hat{p}_{\min}\}\,, \text{ if } x_t = \text{MPS} \end{cases} \tag{8}$$

If we define$\alpha = \lambda = \left(\frac{\hat{p}_{\min}}{0.5}\right)^{\frac{1}{63}}$, then it can be seen that the probability estimation equation (9) is based on equation (3) and the two equations have a similar form. Then if we state that the input symbol x_t=1 is the MPS and

$$x_t\text{=}0$$

is the LPS, the SLWE-based probability estimation for symbol can be defined as:

$$p_{t+1} = \begin{cases} \lambda p_t + (1-\lambda)\,, \text{ if } x_t = \text{LPS} \\ \max\{\lambda p_t,\ p_{\min}\}\,, \text{ if } x_t = \text{MPS} \end{cases} \tag{9}$$

where $\lambda = \left(\frac{p_{\min}}{0.5}\right)^{\frac{1}{63}}$, and according to the binary symbols resulted from quantization, $p_{\min}$ is obtained corresponding value from the table of the initial probability. In the implementation, the interval is expressed in integer data. When the coder updates the range of the encoded symbols, the interval size of the updated symbol, i.e., T, may be less than 1, which may lead to the problem that the leakage of the encoded symbol in the update process. Therefore, we define that if

$$T = R \times p < 1 \tag{10}$$

$$T = 1 \tag{11}$$

where register R satisfies the following inequality:

$$2^{b-2} \le R < 2^{b-1} \tag{12}$$

In the set of $b = 10$, the encoding procedure is shown in Fig.2.

Compared with M-coder, the arithmetic coder using SLWE-based probability estimation requires less memory space and no look-up table, but the initial value of λ based on the different initial probability value. Hence, it is suitable for practical application in video compression applications. Nevertheless, multiplication operation is inevitable in the process of probability estimation withλ. This is not efficient, especially for a hardware implementation [18].

3.2. **Improved SLWE based Probability Estimation.** In order to eliminate the multiplication operation in SLWE-based probability estimation, we design the new expression with the power exponent of 2 as the learning factor, aiming at full use of binary shift operation in the process of the probability estimation. According to the Part A of Section III, the probability p_t shows the probability of MPS at 't' time. Suppose x_t=1 for MPS. Derived from the equation (7), the following equation can be obtained:

$$p_{t+1} = \lambda p_t + (1-\lambda)\ x_t \tag{13}$$

If we define $\lambda = 1 - \frac{1}{2^w}$, and multiply both sides of the equation (14) by 2^w, we achieve:

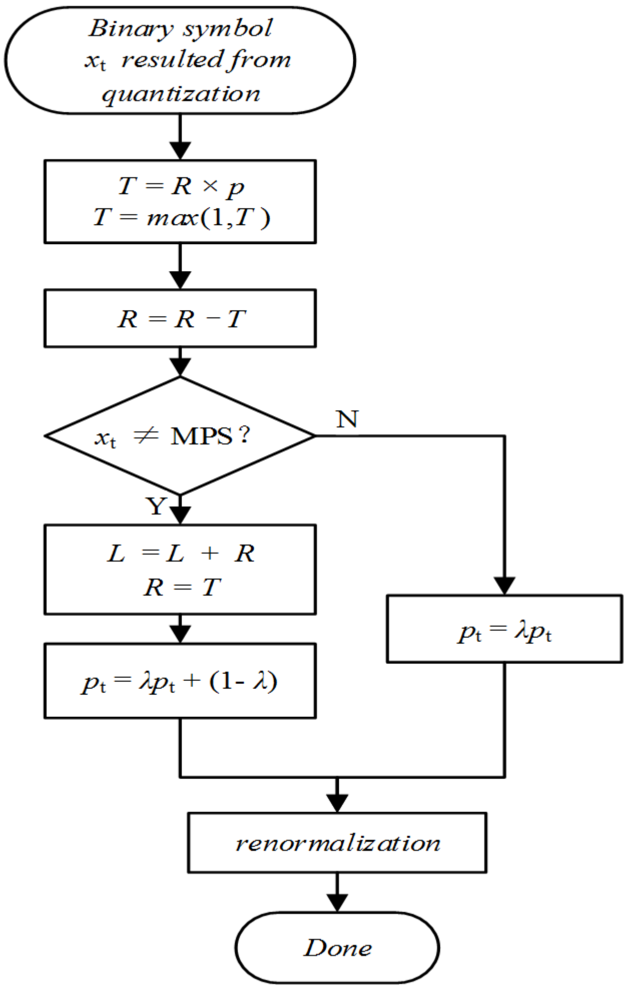

FIGURE 2. Flow of Probability estimation based on SLWE

$$2^w p_{t+1} = \left(1 - \frac{1}{2^w}\right) 2^w p_t + x_t \tag{14}$$

Assume that$s_t = 2^w p_t$, i.e.,$p_t = \frac{s_t}{2^w}$, we can sequentially obtain the equation as follow:

$$s_{t+1} = \begin{cases} s_t - \frac{1}{2^w} s_t + 1 \; x_t = 1 \\ s_t - \frac{1}{2^w} s_t \; x_t = 0 \end{cases} \tag{15}$$

Based on the above calculation process, we can see that s_t update is equivalent to probability p_t update. In other words, the update process of s_t denotes the update process of the probability p_t. Moreover, $s_0$can be obtained when p_0 is known according to $s_t = 2^w p_t$. Thus, multiplication operation with λ in the probability estimation process described in equation (14) is replaced by division operation with 2^w in equation (16). It is worth noting that division operation by 2^w can be conveniently implemented by binary right shifting operation. Accordingly, the estimated probability can be computed by simple operations, including binary shifting, addition and subtraction.

The flow of the aforementioned probability estimation method is demonstrated in Fig.3. It can be seen that the range occupied by the probability p_t, i.e., T, can be obtained by $T = R \times p_t$, and then $T = R \times \frac{s_t}{2^w}$. In the process of probability estimation, neither look-up table nor multiplication operation is required.

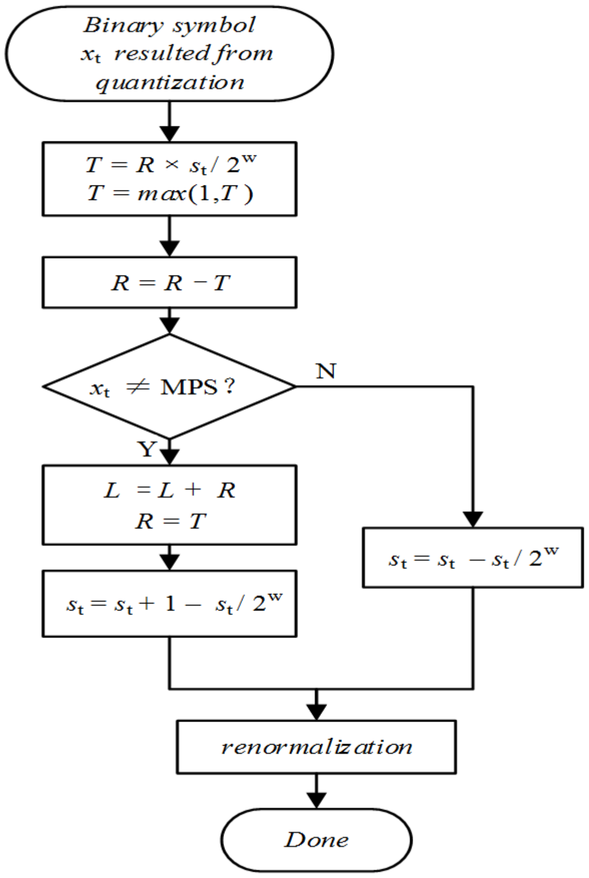

FIGURE 3. Flow of Probability estimation based on the improved SLWE

Furthermore, in order to enhance the precision of the probability estimation, it is necessary to adjust λ based on the characteristics of input symbols instead of fixed λ. In a simple yet effective way, we adaptively update w according to the number of input symbol. Specifically, the update process of $\lambda = 1 - \frac{1}{2^w}$ is shown in Fig.4, where symbol number represents the number of encoded symbols, and w denotes the variable in the expression of$\lambda = 1 - \frac{1}{2^w}$.

When the symbol number reaches a certain number, e.g. 3, 12, 24, w adds 1. Accordingly,λ varies with the change of w. The symbol number returns to zero as soon as it has risen to 64, meanwhile, w will also return to the minimum value. This adaptive process will ensure the best λ with the change of the encoded symbols and the precision of probability estimation.

4. **Experiments and Analysis.**

In order to analyze the performance of the proposed methods, experiments are respectively carried out on synthetic data and standard test video sequences. Using the known characteristics of the synthetic data as the reference, coding redundancy, computation complexity and precision of probability estimation of the proposed method are evaluated in Part A. Furthermore, the practical coding efficiency is evaluated on the standard test video sequences in Part B.

4.1. **Experiments on synthetic data.**

Experiments on synthetic data In common cases, the data to be encoded is non-stationary. For the different stages of data, there is a different optimal λ value to achieve the best probability estimation and coding results. Therefore, it is very important to choose an appropriate $\lambda = 1 - \frac{1}{2^w}$ on different stages of data to be encoded.

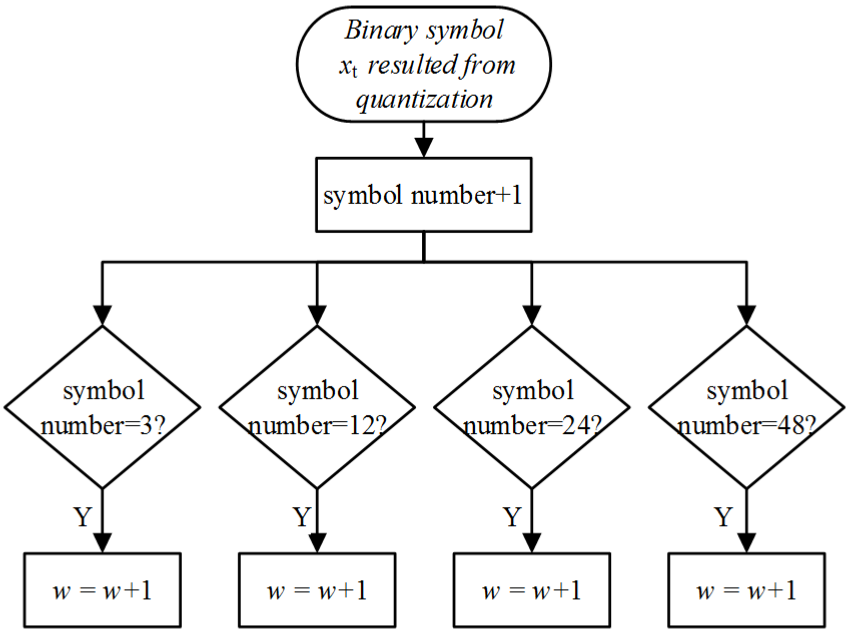

FIGURE 4. Update process of λ

Suppose a stationary binary memory-less source with probability of ones p. The coding redundancy $r(p)$ [18] caused by the probability estimation precision can be defined as follow:

$$r(p) = \sum_i \left(-(1-p) \times \log_2(1-\hat{p}_i) - p \times \log_2 \hat{p}_i\right) - h(p) \tag{16}$$

where $\hat{p}_i$ is a probability estimation value for each stage of the data in the whole coding process, and $h(p)$ is the entropy of the source with probability of ones$p$.

In fact, a coding redundancy also depends on the number of bits for representation of registers L and R. According to the paper [22], the coding redundancy is estimated by the following equation:

$$r(p) = \lim_{N\to\infty}\left(\frac{B}{N}\right) - h(p) \tag{17}$$

where N denotes the number of input binary symbols and B is the size of the compressed bit stream.

According to equation (18), coding redundancy results for the original CABAC and the proposed method are listed in Table I. For both cases, we define the values $b = 10$ and$N = 10^8$.

It can be seen that the coding redundancy for the proposed method depends on the λ, in other words, a larger λ leads to a less redundancy. Compared with the original CABAC, the proposed method exhibits a higher redundancy with $\lambda = 1 - \frac{1}{2^3}$ and a lower redundancy with $\lambda = 1 - \frac{1}{2^9}$ for the most cases. For$\lambda = 1 - \frac{1}{2^6}$, it has a higher redundancy for the low probabilities and a lower redundancy for the high probabilities of ones. Thus, the coding redundancy tends to be lower with the increase of λ.

Computational Complexity Analysis

As shown in Fig.1, Fig.2 and Fig.3, the number of operations in the interval division part of an arithmetic coder is directly proportional to the number of input binary symbols. On the other hand, the number of the used re-normalization is proportional to the number

TABLE 1. Coding redundancy $r(p)$ for the original CABAC and the proposed method

P	Original CABAC	Proposed method		
		$\lambda=1-\frac{1}{2^3}$	$\lambda=1-\frac{1}{2^6}$	$\lambda=1-\frac{1}{2^9}$
0	0.029	0.0042	0.0042	0.0042
10^{-5}	0.0287	0.0039	0.0039	0.0039
10^{-4}	0.0281	0.0036	0.0034	0.0034
10^{-3}	0.0239	0.0028	0.0024	0.0019
10^{-2}	0.0102	0.014	0.0081	0.0064
0.02	0.008	0.028	0.019	0.008
0.03	0.009	0.034	0.016	0.007
0.04	0.012	0.037	0.017	0.008
0.06	0.017	0.036	0.016	0.006
0.08	0.02	0.031	0.015	0.007
0.1	0.021	0.028	0.014	0.006
0.2	0.021	0.025	0.013	0.007
0.3	0.022	0.028	0.014	0.007
0.4	0.02	0.026	0.013	0.008
0.5	0.02	0.02	0.01	0

of bits in the output bit stream. Define N as the number of the input binary symbols, R as the size of the output bit stream. The complexity per input symbol (denoted as C) for the proposed method can be formulated as the following equation [23]:

$$C = \frac{\alpha N + \beta R}{N} \tag{18}$$

where α denotes the computation complexity of the interval division part per input binary symbol and β represents the computation complexity of the renormalization part per output binary symbol. For arithmetic coder, the output bit stream size is:

$$R = N \times (h(p) + r(p)) \tag{19}$$

Therefore, the complexity per input symbol is a linear function of the source entropy and coding redundancy which can be described as:

$$C(p) = \alpha + \beta(h(p) + r(p)) \tag{20}$$

According to equation (21), it can be seen that the complexity $C(p)$ is low for zero-entropy source (i.e. $h(p) = 0$). When $h(p)$ is zero and $r(p)$ is invariable and very small, the impact of R on $C(p)$ is negligible. As the complexity of the interval division part per input binary symbol, the complexity $C(p)$ is mainly determined by α. In addition, β imposes greater impact with the increase of $h(p)$ as β is the computation complexity of the re-normalization part per output binary symbol. For the invariable $r(p)$, the complexity also reaches the maximum when $h(p) = 1$. From equation (21), if except for the coding redundancy $r(p)$ each other parameter remains unchanged, the coding scheme will have a high complexity with a high coding redundancy.

Table II presents the complexity $C(p)$ for the original CABAC and the proposed method. According to the papers [24] and [25], we measure the complexity in terms of CPU cycles. In the case of the each stated probability, we generated $N = 10^8$ input

Table 2. CPU cycles pre symbol of processor for the original CABAC and the proposed method

P	Original CABAC	Proposed method $\lambda = 1 - \frac{1}{2^w}$				average gain (%)
		w =3	w=6	w=9	average	
0	16.3	14.2	14.2	14.2	14.2	12.9
0.05	22.5	22.7	22.1	21.8	22.2	1.3
0.1	27.4	27.7	26.6	26.3	26.8	2.2
0.2	34.5	34.1	33.5	32.6	33.4	3.2
0.3	41.2	41.1	39.4	38.3	39.6	3.9
0.4	45.6	45.6	44.8	44.6	45.0	1.3
0.5	48.3	48.5	47.8	47.4	47.9	0.8

binary symbols. The CPU cycles of Processor Intel Core i3M 2.1 GHz are measured by the Average CPU Cycles software [26].

According to Table II, we can observe that the complexity for the proposed method depends on the learning factor λ ($\lambda = 1 - \frac{1}{2^w}$) which determines different coding redundancies. Larger learning factor λ provides less redundancy and less complexity as shown in equation (21). Besides, the proposed method has up to 12.9% less computation complexity for the almost zero probability. For the rest of probabilities, the complexity difference ranges from 0.5% to 4.0%. It implies that compared with the original CABAC the coder using the proposed method has comparable computation complexity in these cases.

Analysis for Precision of Probability Estimation

Taking the known probabilities distribution of the given synthetic data into consideration, we can obtain the estimated probabilities by the proposed method and compute the probability estimation precision by

$$preci = \text{1-}|\frac{p - \hat{p}}{p}| \times 100 \qquad (21)$$

where p denotes the real probability of ones, $\hat{p}$ is the estimated probability of ones and $preci$ represents the probability estimation precision.

Table III provides the probability estimation precision for the original CABAC and the proposed method. When the real probability of ones is zero, the estimated probability values are same as the true probability values. Meanwhile, when real probability is set to other values, the estimated probability values are closer to the real probability value with the growing of λ. Specifically, the probability estimation precision ranges from 94.5% to 97.5%.

4.2. **Experiments on standard test video sequence.** To obtain practical results, the proposed method was embedded into HM 3.0 [27] as the reference software of the HEVC standard. HM codec was running with the low-delay configuration.

According to the reference [18], the initial state for each binary context is defined as:

TABLE 3. Probability estimation precision of the original CABAC and the proposed method

P	Proposed method $\lambda=1-\frac{1}{2^w}$				Precision (%)
	w=3	w=6	w=9	average	
0	0	0	0	0	100
0.05	0.046	0.047	0.048	0.047	94.5
0.1	0.091	0.096	0.097	0.095	95.1
0.2	0.214	0.208	0.202	0.208	95.8
0.3	0.317	0.310	0.305	0.311	96.3
0.4	0.416	0.415	0.411	0.414	96.9
0.5	0.519	0.512	0.506	0.513	97.5

$$s_0 = \max\left\{0.5 - \frac{1}{2^w},\ [2^w\hat{p}_0 + 0.5]\right\} \tag{22}$$

where $\hat{p}_0$ is the initial probability predefined in the standard. When the coder initialized, the corresponding initialization probability is given by the standard according to the different state of the binary data resulting from the quantization.

In the experiments, the results of quantization are input as the non-stationary source. With the different quantization parameter (QP), the experiments are carried on standard test video sequence in terms of computation speed. The original CABAC and the proposed methods generated identical reconstructed video for variable QPs.

Practical results are obtained for the first 60 frames of the standard test video sequences [28] [29] with different frame resolutions, including 832×480 (”BasketballDrill”), (”PartyScene”), (”BQMall”), 416×240 (”RaceHorses”), (”BasketballPass”).

Using different QPs, Fig.5 demonstrates that the average computation time for all the standard test video sequences. There is negligible impact on encoding time. TABLE III shows the value of computation time as the following equation:

$$TI = \frac{Tp - To}{Tp} \times 100 \tag{23}$$

where Tp and To denote the encoding times of the proposed method and the original CABAC and TI is the values increased slightly comparison the proposed method and the original CABAC.

The encoding results of the proposed method are compared with the original CABAC. To compare the performance of the two approaches, two criteria are utilised: average PSNR difference (Bjntegaard Delta PSNR; in decibels) and average bit rate difference (Bjntegaard Delta BR; as a percentage), as defined in [30-31]. TABLE IV shows the simulation results of the original CABAC and the proposed method for seven standard test video sequences and QP values ranging from 24 to 40. The GOP size for the hierarchical B-frame structure was set to 8. TABLE IV shows the overall average results of BDPSNR = 0.4176dB and BDBR = -2.11%. This indicates that (1) with the same bit rate, the proposed scheme increases the average PSNR by 0.4176dB, and (2) with the same video quality, the proposed algorithm produces a bit rate saving of 2.11% compared with the original CABAC. In general, the PSNR of each of the four sequences is increased at

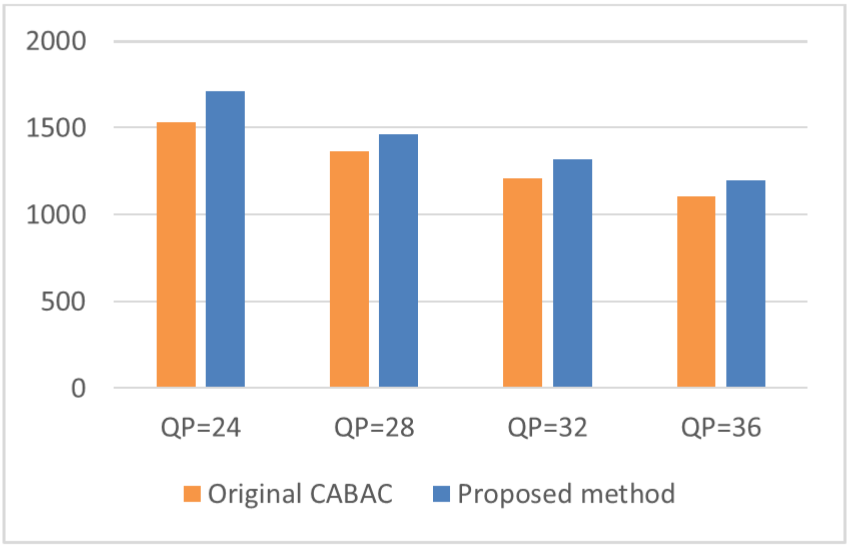

FIGURE 5. Comparison of the average computation speed (in sec) between original CABAC and the proposed method in CABAC of HEVC standard

TABLE 4. Computation speed performance compared to original CABAC for HEVC standard in case of SLWE

Sequence	QP	Original CABAC	Proposed method	*T* (%)	Average (%)
		To (sec)	*Tp* (sec)		
Basketball Drill	24	2015	2154	6.45	7.93
	28	1806	1914	5.64	
	32	1562	1727	9.55	
	36	1430	1590	10.06	
PartyScen e	24	2507	3047	17.72	9.83
	28	2231	2398	6.96	
	32	1953	2106	7.27	
	36	1736	1874	6.71	
BQMall	24	2324	2474	6.06	8.01
	28	2072	2236	7.33	
	32	1888	2077	9.09	
	36	1718	1899	9.53	
RaceHorse s	24	320	352	9.09	5.86
	28	286	306	6.53	
	32	258	269	4.09	
	36	232	241	3.73	
Basketball Pass	24	500	516	3.10	6.39
	28	435	463	6.05	
	32	387	421	8.08	
	36	351	383	8.36	

Table 5. Rate distortion performance compared to original CABAC for HEVC standard in case of SLWE

Sequence	QP	Original CABAC		Proposed method		BDBR (%)	BD PSNR (dB)
		BR (kbits/s)	PSNR (dB)	BR (kbits/s)	PSNR (dB)		
BasketballDrill	24	16437.5	41.08	16523.6	41.05	-2.74	0.600
	28	9966	38.51	10030.8	38.49		
	32	6076	36.27	6111.5	36.26		
	36	3709.9	34.22	3730	34.21		
PartyScene	24	36557.1	39.95	36631.4	39.93	-1.85	0.273
	28	24634.1	36.87	24689.6	36.86		
	32	16228.1	34.07	16282.5	34.05		
	36	9988.8	31.3	10021.5	31.27		
BQMall	24	19117.6	41.54	19152.3	41.53	-1.25	0.233
	28	12575.6	39.22	12603.8	39.2		
	32	8223	36.91	8243.2	36.1		
	36	5235.3	34.57	5249.7	33.53		
RaceHorses	24	3696.1	41.18	3692.0	41.16	-3.15	0.709
	28	2397.0	38.14	2435.7	38.15		
	32	1476.3	35.35	1477.1	35.33		
	36	842.6	32.81	844.2	32.78		
BasketballPass	24	4370.3	42.08	4377.1	42.05	-1.56	0.273
	28	2839.4	39.29	2845.9	39.28		
	32	1818.1	36.75	1824	36.73		
	36	1129.3	34.34	1133.9	34.31		
Average						-2.11	0.4176

all ranges of target bit rate. Therefore, it improves the coding quality and the coding efficiency and this is true regardless of target bit rate.

5. Conclusion.

An efficiency adaptive probability estimation method was presented by improved SLWE without multiplication and look-up table. Accordingly, it provides a simple mechanism to achieve the tradeoff between the speed of probability adaption and the compression efficiency and quality of compressed data with the new expression of learning factor and update method. Meanwhile, the proposed method does not require any changes in the context-modeling and can be easily embedded into any compression scheme using binary arithmetic coding. It allows a reduction in memory space in software and hardware implementations. Future work will focus on the improvement of the precision of probability estimation and the practicability in screen content video coding.

Acknowledgment. This work was supported in part by a grant from the National High Technology Research and Development Program of China (863 Program) (No. 2012AA12A405), the National Science Foundation of China (No.61102159), Natural Science Foundation of Heilongjiang Province, China(Grant No.LC2016025).

REFERENCES

[1] K. Xiong, C. Wei, Multiple Model Adaptive Estimator for Nonlinear System with Unknown Disturbance, *Asian Journal of Control*, 2016.

[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. 620636, 2003.

[3] V. Sze and M. Budagavi, Overview of the high efficiency video coding (HEVC) standard, *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 16491668, 2012.

[4] Digital compression and coding of continuous-tone still images, *ITU-T and ISO/IEC JTC1*, 1992.

[5] JPEG 2000 Image Coding System: Core Coding System, *ITU-T and ISO/IEC JTC 1, ITU-T Rec. T.800 and ISO/IEC 15444-1*, 2000.

[6] H. Eeckhaut, B. Schrauwen, M. Christiaens, and J. Van Campenhout, Tuning the M-coder to improve dirac's entropy coding, *WSEAS Trans. Inf. Sci. Applicat.*, pp. 15631571, 2005

[7] D. L. Duttweiler, C. Chamzas, Probability estimation in arithmetic and adaptive-Huffman entropy coders, *IEEE Transactions on Image Processing*, vol. 4, no.3, pp. 237-246, 1995,

[8] N. Merhav and M. Feder, A strong version of the redundancy-capacity theorem of universal coding, *IEEE Trans. Inform. Theory*, vol. 41, pp. 714722, 1995

[9] R. G. Gallager, Variations on a theme by Huffman, *IEEE Tran. Info Theory,* vol. 24, no.6, pp.668-674, 1978

[10] P. Sunehag, W. Shao, M. Hutter, Coding of non-stationary sources as a foundation for detecting change points and outliers in binary time-series, *Proc. Tenth Australasian Data Mining Conferenc*, vol. 134. pp. 79-84, 2012.

[11] B. J. Wang, Y. Zhan, A survey and performance evaluation on sliding window for data stream, *Communication Software and Networks (ICCSN), IEEE 3rd International Conference*, pp. 654 657, 2011

[12] E. Belyaev, M. Gilmutdinov, and A. Turlikov, Binary arithmetic coding system with adaptive probability estimation by Virtual Sliding Window, *Proc. 10th IEEE Int. Symp. Consumer Electron*, pp. 15, 2006.

[13] T. Leighton and R. L. Rivest, Estimating a probability using finite memory, *IEEE Trans. Inform. Theory,* vol. IT-32, pp. 733742, 1986.

[14] B.Y. Ryabko, Imaginary sliding window as a tool for data compression, *Probl. Inform. Transm.*, pp. 156163, 1996.

[15] E. Meron and M. Feder, Finite-Memory Universal Prediction of Individual Sequences, *IEEE Trans. Inform. Theory*, vol. 50, n. 7, pp. 15061523, 2004.

[16] B. J. Oommen, L. Rueda L, A new family of weak estimators for training in non-stationary distributions, *Structural, Syntactic, and Statistical Pattern Recognition. Springer*, pp.644-652,2004 .

[17] B. J. Oommen, L. Rueda L. Stochastic learning-based weak estimation of multinomial random variables and its applications to pattern recognition in non-stationary environments, vol.39, no.3, pp.328-341, 2006.

[18] B. J. Oommen, L. Rueda L, On utilizing stochastic learning weak estimators for training and classification of patterns with non-stationary distributions, *Advances in Artificial Intelligence Lecture Notes in Computer Science. Springer*, pp.107-120, 2005.

[19] B. Evgeny, T. Andrey, E. Karen, Senior and Moncef Gabbouj, An Efficient Adaptive Binary Arithmetic Coder With Low Memory Requirement, IEEE Journal of Selected Topics in Signal Processing, vol. 7, no. 6, 2013

[20] A. Moffat, R. Neal, and I. Witten, Arithmetic coding revisited, *ACM Trans. Inf. Syst.*, vol. 16, n. 3, pp. 256294, 1998.

[21] E. Krichevski and V. Trofimov, The performance of universal encoding, *IEEE Trans. Inf. Theory,* vol. 27, n. 2, pp. 199207, 1981.

[22] D. Marpe and T. Wiegand, A highly efficient multiplication-free binary arithmetic coder and its application in video coding, *Proc. IEEE Int. Conf. Image Proc.*, Vol. 2, pp. 263266, 2003.

[23] B. Y. Ryabko and A. N. Fionov, An efficient method for adaptive arithmetic coding of sources with large alphabets, *Prob. Inf. Trans.*, vol. 35, no. 4, pp. 95108, 1999.

[24] E. Belyaev, A. Veselov, A. Turlikov, and K. Liu, Complexity analysis of adaptive binary arithmetic coding software implementations, *Proc. 11th Int. Conf. Next Gen. Wired/Wireless Adv. Netw.*, 2011.

[25] C. Grecos and M. Yang, Fast inter mode prediction for P slices in the H264 video coding standard, IEEE Trans. Broadcasting, vol. 51, no. 2, pp. 256253, Jun. 2005.

[26] W. Lee, H. Choi, and S. Wonyong, Fast block mode decision for H.264/AVC on a programmable digital signal processor, *Proc. IEEE Workshop Signal Process. Syst.*, pp. 169174, 2007.

[27] http://user.tninet.se/ jad615g/averagecpu/AverageCPUcycles.exe

[28] High Efficiency Video Coding, [Online]. Available: http://www.h265.net/

[29] MVC Test Sequences. [Online]. Available: http://www.merl.com/pub/avetro/mvc-testseq/orig-yuv/

[30] Xiph.org Test Media. [Online]. Available: http://media.xiph.org/video/derf/

[31] G. Bjontegaard, Calculation of average PSNR differences between RDcurves, ITU-T VCEG, 13th VCEG Meeting, Doc. VCEG-M33 Std., 2001.

[32] X. Lu, and G. R. Martin, Fast Mode Decision Algorithm for the H.264/AVC Scalable Video Coding Extension, Circuits & Systems for Video Technology IEEE Transactions, vol. 23, n. 5, pp. 846-855, 2013