Improved Efficiency on Adaptive Arithmetic Coding for Data Compression Using Range-Adjusting Scheme, Increasingly Adjusting Step, and Mutual-Learning Scheme

Jian-Jiun Ding, I-Hsiang Wang, and Hung-Yi Chen

***Abstract*—Context-based adaptive arithmetic coding (CAAC) has high coding efficiency and is adopted by the majority of advanced compression algorithms. In this paper, five new techniques are proposed to further improve the performance of CAAC. They make the frequency table (the table used to estimate the probability distribution of data according to the past input) of CAAC converge to the true probability distribution rapidly and hence improve the coding efficiency. Instead of varying only one entry of the frequency table, the proposed range-adjusting scheme adjusts the entries near to the current input value together. With the proposed mutual-learning scheme, the frequency tables of the contexts highly correlated to the current context are also adjusted. The proposed increasingly adjusting step scheme applies a greater adjusting step for recent data. The proposed adaptive initialization scheme uses a proper model to initialize the frequency table. Moreover, a local frequency table is generated according to local information. We perform several simulations on edge-directed prediction (EDP) based lossless image compression, coefficient encoding in JPEG, bit plane coding in JPEG 2000, and motion vector residue coding in video compression. All simulations confirm that the proposed techniques can reduce the bit rate and are beneficial for data compression.**

***Index Terms*—Data compression; adaptive arithmetic coding; context-based adaptive arithmetic coding; entropy coding; lossless image compression by edge-directed prediction**

I. INTRODUCTION

Entropy coding, including Huffman coding [1], Pareto coding [2], Golomb-Rice coding [3, 4], static arithmetic coding [5], adaptive arithmetic coding [5-10], context-based adaptive binary arithmetic coding (CABAC) [8], and context-based adaptive arithmetic coding (CAAC) [9, 10], play an important role in data compression. Huffman coding is adopted in some compression standards, including JPEG [11]. However, for the Huffman code, to achieve the optimal result, the explicit probability distribution of the input data must be known. In [2-4], they used the Pareto or geometric distribution to model the probability distribution of the input data. However, as in the Huffman code, because they encode each input separately, the coding efficiency may not be as effective as that of arithmetic coding.

Arithmetic coding encodes the input data jointly. There are two types of arithmetic coding: static arithmetic coding, and adaptive arithmetic coding. When performing static arithmetic coding (SAC), one first records the probability distribution of the input data and initializes the probability range [L, H] to [0, 1]. Then, L and the H are adjusted iteratively according to the input data (denoted by X). If the current value of X is S, then

$$L_{(new)} = L + (H-L)\cdot\sum_{v=v_0}^{s-1} p\left(X=v\right),$$

$$H_{(new)} = L_{(new)} + (H-L)\cdot p\left(X=s\right) \qquad (1)$$

where $p(X = v)$ is the estimated probability of $X = v$ and v_0 is the minimal possible value of X. Finally, a series of bits is used to represent the range [L, H]. SAC can achieve higher efficiency than Huffman coding. However, it requires extra bits to record the estimated probability of the input data.

Adaptive arithmetic coding (AAC) does not require coding the probability distribution of the input data in advance; rather, it applies a frequency table (denoted by $F[v]$). The value of $F[v]$ reflects the probability of $X = v$ where X is the input data, and v is the possible value of the input (for example, when coding a binary stream, v can be "0" or "1"). Initially, $F[v]$ is set to "1" for all v. Then, it is adjusted iteratively by

$$F[v] \Leftarrow F[v]+1 \quad \text{if } X[n] = v. \qquad (2)$$

Note that, if $X[m] = v$ (where $m < n$) occurs k times, then the value of $F[v]$ is k+1 when encoding the n^{th} input. $F[v]$/sum($F[v]$) approaches the true probability distribution of the input data as n increases:

$$F[v]/\text{sum}(F[v]) \cong P(x[n] = v) \quad \text{when } n \text{ is very large}$$

where $P(\,)$ is the probability. Then, we apply the frequency table to adjust the upper and the lower bounds of the probability range (denoted by [L, H]) and encode [L, H] in a binary form.

Manuscript received in May 2017 and accepted in Aug. 2017. This work was supported by the National Science Council, R.O.C., under the contract 104-2221-E-002-097-MY3.

Copyright (c) 2017 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending an email to pubs-permissions@ieee.org.

The authors are with the Graduate Institute of Communication Engineering, National Taiwan University, Taipei, Taiwan, 10617, R.O.C. (e-mail: jjding@ntu.edu.tw, r05942042@ntu.edu.tw, r03942139@ntu.edu.tw)

When performing decoding, it is not necessary to the frequency table and L and H because when decoding $X[n]$, the values of L, H, and the frequency table can be fully determined by the data that has been decoded (i.e., $X[1]$, $X[2]$, …., $X[n-1]$). Because no extra data is required when retrieving the original data, AAC can achieve even higher coding efficiency than SAC.

The performance of AAC can be further improved by context modeling. CAAC [9, 10] applies different frequency tables $F_C[v]$ for different contexts C and the context value C is determined from the causal part (i.e., $X[n_1]$ where $n_1 \leq n-1$). A simplified flowchart of CAAC is presented in Fig. 1. CAAC has been widely used for text compression [12] and lossy, nearly lossless, and lossless image compression [13-17]. Moreover, there are some advanced coding algorithms that have some difference from CAAC. For example, the MQ coder, which is the entropy coding method used in JPEG2000 [18-20], can be viewed as an intermediate of SAC and CAAC. It uses table switching to adapt to the input signal and is free of multiplication. Moreover, CABAC [8] was developed for video coding and is used in the HEVC standard.

The goal of this paper is to further improve the coding efficiency of AAC and CAAC. Although AAC and CAAC achieve very high compression ratios, we believe that their compression performances can be further improved. For example, in Fig. 1, if $X[n] = S$ and the context is C, only the value of $F_C[v]$ in the frequency table is adjusted. However, the probability of $X = v$ is always highly correlated to that of $X = v+\tau$ if τ is small. Moreover, if the context C_1 is closely related to the context C, their corresponding probability distributions should be similar. Therefore, rather than the method plotted in Fig. 1, there can be a superior method to adjust the frequency table.

In Section II, the five proposed techniques to improve the performances of AAC and CAAC are introduced in detail. These techniques can make the frequency table converge to the true probability distribution in a very short time. In Section III, we provide several examples that use the proposed coding algorithm for (i) edge-directed prediction (EDP) lossless image compression, (ii)(iii) encoding the DC and AC terms in JPEG, (iv) bit plane coding in the JPEG2000 process, and (v) encoding the residue of motion vector estimation in video compression. All examples demonstrate that the proposed techniques can further improve the coding efficiencies of AAC and CAAC and are very beneficial for data compression.

II. Improvement for Adaptive Arithmetic Coding

AAC and CAAC are powerful for entropy coding and are adopted by the majority of advanced compression algorithms. In this section, we propose five techniques, including (a) initialization of the frequency table, (b) the range-adjusting scheme, (c) the increasingly adjusting step, (d) the mutual-learning scheme, and (e) the local frequency table, to further improve the coding efficiency of AAC and CAAC. Each of them will be illustrated in the following subsections.

A. Initialization of the Frequency Table

In the conventional adaptive arithmetic coding algorithm, the frequency table is initialized by $F[v] = 1$ for all v. However, this is far from the true distribution of the input data. Therefore,

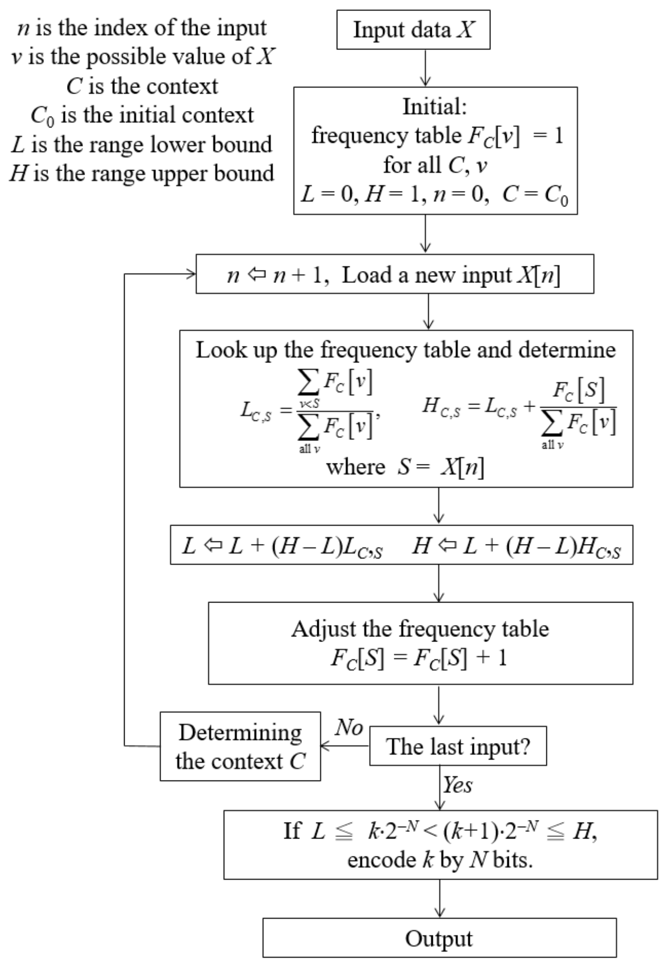

n is the index of the input
v is the possible value of X
C is the context
C_0 is the initial context
L is the range lower bound
H is the range upper bound
Input data X
Initial: frequency table $F_C[v] = 1$ for all C, v; $L = 0, H = 1, n = 0, C = C_0$
$n \Leftarrow n+1$, Load a new input $X[n]$
Look up the frequency table and determine $L_{C,S} = \frac{\sum_{v<S} F_C[v]}{\sum_{\text{all } v} F_C[v]}$, $H_{C,S} = L_{C,S} + \frac{F_C[S]}{\sum_{\text{all } v} F_C[v]}$ where $S = X[n]$
$L \Leftarrow L + (H-L)L_{C,S}$ $H \Leftarrow L + (H-L)H_{C,S}$
Adjust the frequency table $F_C[S] = F_C[S] + 1$
The last input?
No
Determining the context C
Yes
If $L \leqq k \cdot 2^{-N} < (k+1) \cdot 2^{-N} \leqq H$, encode k by N bits.
Output

Fig. 1 A simplified flowchart of the original context-based adaptive arithmetic coding (CAAC) process.

we use another method to initialize the frequency table, which can be viewed an intermediate of the initializing methods of SAC (recording the probability distribution of the input in advance) and AAC (setting $F[v] = 1$ for all v). That is, we initialize the frequency table using some probability model, such as a Gaussian model

$$F[v] = \max\left(a_1 \exp\left(-\sigma_1 v^2\right), \varepsilon\right) \tag{3}$$

or an exponential model

$$F[v] = \max\left(a_2 \exp\left(-\sigma_2 v\right), \varepsilon\right). \tag{4}$$

where a_1 or a_2 controls the maximal value of the initial frequency table, ε controls the minimum of the initial frequency table and σ_1 or σ_2 controls the decay rate. Compared to setting $F[v] = 1$ for all n, using (3) and (4) can better approximate the probability distribution of the input data and hence achieve superior compression performance, especially for the first several inputs.

In (3) and (4), it is preferable to set a_1, a_2, σ_1, and σ_2 properly to allow $F[v]$ to effectively approximate the average probability distribution of the input data. ε can be chosen as a considerable small number, however, it cannot be zero. Otherwise, it could happen that $\log_2 F[v] \to -\infty$ and the Kullback–Leibler distance between $F[v]$/sum($F[v]$) and the true probability distribution would become large.

B. Range-Adjusting Scheme

In the original AAC algorithm, if the n^{th} input is $X[n] = v$, then, in the frequency table, only the value of $F[v]$ is adjusted, as in (2). However, we think that the probability of $P(X[n] = v)$ has very high correlation with $P(X[n] = v_1)$ when v_1 is near to v. Therefore, if $X[n] = v$, then in addition to adjusting $F[v]$, the value of $F[v_1]$ should also be adjusted when $|v_1 - v|$ is small. We call this the *range-adjusting scheme*.

For example, if X is the DC difference of the 8×8 blocks in an image, when one finds that the DC difference between two blocks is 10, then in the frequency table, not only the value of $F[10]$ but also the values of $F[8]$, $F[9]$, $F[11]$, and $F[12]$ should be adjusted, because $P(X = v_1)$ and $P(X = 10)$ have very high correlation if v_1 is close to 10.

Based on this concept, instead of (2), we apply the following method to adjust the frequency table:

$$F[v_1] \Leftarrow F[v_1] + A{\cdot}B[v, v_1] \quad \text{if } X[n] = v \tag{5}$$

where A is some constant and $B[v, v_1]$ is a function that has the following properties:

(i) $B[v, v_1]$ decreases with the difference between v and v_1,

(ii) $\sum_{v_1} B[v, v_1] = 1.$ (6)

There are several possible choices for $B[v, v_1]$. For example, one can choose $B[v, v_1]$ as a *rectangular function form*:

$$B[v, v_1] = 1/b \quad \text{if } v_2 \le v_1 \le v_3,\ B[v, v_1] = 0 \quad \text{otherwise}, \tag{7}$$

$$v_2 = \text{ceil}(\mu{\cdot}v), \quad v_3 = \text{floor}(v/\mu),$$

$$\mu \text{ is a constant and } 0 < \mu \le 1, \quad b = v_3 - v_2 + 1 \tag{8}$$

where "ceil" implies rounding towards infinity and "floor" implies rounding towards minus infinity. Then, (5) becomes

$$F[v_1] \Leftarrow F[v_1] + A/c \quad \text{if } X[n] = v \text{ and } v_2 \le v_1 \le v_3. \tag{9}$$

Moreover, one can also set $B[v, v_1]$ as a *triangular function form*:

$$B[v,v_1] = c(v_1 - v_2)/(v - v_2) \quad \text{if } v_2 \le v_1 \le v,$$
$$B[v,v_1] = c(v_3 - v_1)/(v_3 - v) \quad \text{if } v < v_1 \le v_3,$$
$$B[v, v_1] = 0 \quad \text{otherwise} \tag{10}$$

where v_2 and v_3 are defined the same as in (8) and

$$c = 1\bigg/\left(\sum_{v_1=v_2}^{v} \frac{v_1 - v_2}{v - v_2} + \sum_{v_1=v+1}^{v_3} \frac{v_3 - v_1}{v_3 - v}\right). \tag{11}$$

Furthermore, $B[v, v_1]$ can also be chosen as an *exponential function form*:

$$B[v,v_1] = c_1 \exp\left(-\sigma |v - v_1|^{\alpha}\right) \tag{12}$$

where α and σ are positive constants and

$$c_1 = 1/\sum_{v_1} \exp\left(-\sigma |v - v_1|^{\alpha}\right).$$

One can also set $B[v, v_1]$ as a *normalized exponential function form*:

$$B[v,v_1] = c_2 \exp\left(-\sigma |1 - v_1/v|^{\alpha}\right)$$

$$\text{where } c_2 = 1/\sum_{v_1} \exp\left(-\sigma |1 - v_1/v|^{\alpha}\right). \tag{13}$$

If (13) is applied, then the formula to adjust the frequency table becomes:

$$F[v_1] \Leftarrow F[v_1] + c_2 A \cdot \exp\left(-\sigma |1 - v_1/v|^{\alpha}\right) \quad \text{if } X[n] = v. \tag{14}$$

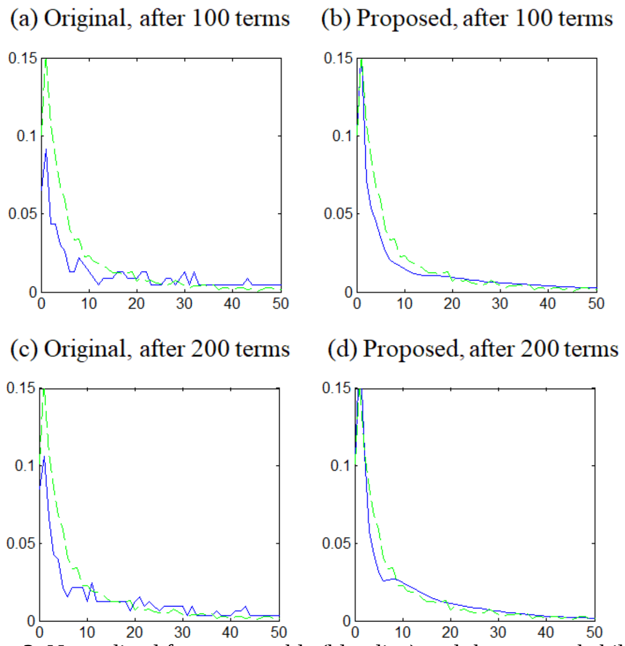


Fig. 2 Normalized frequency table (blue line) and the true probability distribution (green dash line) when encoding the DC differences of Peppers. (a)(c) Using the original method in (2) when 100 (or 200) DC difference terms are encoded. (b)(d) Using the proposed range-adjusting scheme when 100 (or 200) terms are encoded. The results indicate that, when using the proposed algorithm, the frequency table can effectively approximate the true probability distribution even if only a small part of the input is encoded.

Note that, when using the original adjusting method in (2), if $X[n] = v$, only the value of $F[v]$ is increased. Conversely, when applying the proposed *range-adjusting scheme*, the value of $F[v_1]$ is also adjusted even if $v_1 \ne v$.

Moreover, from (7), (10), (12), and (13), one can see that the adjusting step $B[v, v_1]$ is large when v_1 is near to v and $B[v, v_1]$ is small when v_1 is far from v.

For example, in Fig. 2, we display the frequency tables (with normalization) using the original adjusting method in (2) and the proposed range-adjusting scheme when encoding the DC difference in the JPEG process. In Figs. 2(b)(d), $B[v, v_1]$ is chosen as the normalized exponential function form as in (13). The parameters are the same as those described in Section III-B. From Figs. 2 (b)(d), one can see that, when using the proposed range-adjusting scheme, the frequency table matches the true probability distribution well even if only a small portion of the DC differences are encoded.

The original AAC has acceptable performance only when the input data is sufficiently long. If the input is overly short, the frequency table will deviate from the true probability distribution. Conversely, when using the proposed range-adjusting scheme, because the frequency table converges to the true probability rapidly, the coding performance is effective even if the input data is short or there are many contexts

The choice of $B[v, v_1]$ should reflect the correlation between $P(X = v)$ and $P(X = v_1)$. Therefore, $B[v, v_1]$ should decrease with $|v - v_1|$. Moreover, if the correlation between $P(X = v)$ and $P(X = v_1)$ decays slowly with $|v - v_1|$, then the parameter b in (7) $v - v_2$ and $v_3 - v$ in (10) should be greater and the value of σ and α in (12) and (13) should be less. Conversely, if the correlation between $P(X = v)$ and $P(X = v_1)$ decays rapidly with $|v - v_1|$, then b, $v - v_2$, and $v_3 - v$ in (10) should be less and σ and α in (12) and (13) should be greater. Normally, the rectangular and triangular forms are easier to implement; however, the exponential and normalized exponential forms yield superior results.

In Section III, we will illustrate that, with the range-adjusting scheme, the coding efficiency of the adaptive arithmetic code can be further improved, *especially for cases that have many contexts* (when there are many contexts, the number of data for each context is small).

C. *Increasing the Adjusting Step Size*

Furthermore, we think that the adjusting step should be increased during the AAC process. For the conventional AAC method in (2), the value of $F[v]$ is incremented by 1 if $X[n] = v$, i.e., the adjusting step is fixed to 1. However, it does not consider the importance of the recent input data. When encoding the nth input data, the information of the $(n - k_1)$th input should be more important than that of the $(n - k_2)$th input if $k_1 < k_2$. However, when using the method in (2), the $(m - k_1)$th input and the $(m - k_2)$th input have equivalent effects on the frequency table because their corresponding adjusting steps are both 1.

Therefore, instead of (2), we suggest that the adjusting step should increase with n:

$$F[v] \Leftarrow F[v] + A[n] \quad \text{if } X[n] = v \qquad (15)$$

where $A[n]$ is a function growing with n. For example, one can set $A[n]$ as the *linear form*:

$$A[n] = a + bn \qquad (16)$$

where a and b are some positive constants. Moreover, one can also set $A[n]$ as the *geometric form*:

$$A[n] = d \cdot c^n \qquad (17)$$

where $d > 0$ and $c > 1$. Note that, if one uses the increasing step size as in (16) and (17), then

$$A[n-k_1] > A[n-k_2] \quad \text{if} \quad k_1 < k_2. \qquad (18)$$

This means that when encoding the nth input data the $(n - k_1)$th input has a more important role than the $(n - k_2)$th input if $k_1 < k_2$. The linear form in (16) is easy to implement and its performance is similar to that of the geometric form in (17). However, when encoding a long sequence of data, the geometric form generally demonstrates considerably superior performance. For the geometric form, the recent input data has an even greater influence on the frequency table. Therefore, the former is more suitable for the short input case and the latter is more suitable for the long input case.

Note that, in (16) and (17), the relative importance of the $(n - k_1)$th input to the $(n - k_2)$th input is

$$\frac{a+bk_1}{a+bk_2} \quad \text{and} \quad c^{k_1-k_2}, \qquad (19)$$

respectively. Therefore, in practical applications, to avoid the over-stressing of the recent input data, one can set c in (17) to be a little larger than 1 (or set b in (16) to be slightly greater than 0). If the correlation between the nth input and the $(n - k)$th input decays rapidly with k, then b and c in (16) and (17) are set to larger values. Otherwise, b is set to be slightly greater than 0 and c should be a little larger than 1. Moreover, if the sum of the initial frequency table is large, a and d in (16) and (17) should be chosen larger.

Moreover, to avoid the frequency table and the adjusting step being overly large, a scale-down technique is applied. That is, if $\sum F[v]$ exceeds a given threshold, one can use the following scaling and quantization operations to adjust the frequency table and the adjusting step:

$$F[v] \Leftarrow Max(F[v]/2, \delta)$$
$$\text{and } A[v] \Leftarrow A[v]/2 \quad \text{if } \sum F[v] > T. \qquad (20)$$

With this mechanism, it can be assured that $F(v)/\text{sum}(F(v))$ is never less than δ/T.

One can further combine the techniques of the increasingly adjusting step and the range-adjusting scheme by modifying the formula of frequency table adjusting as

$$F[v_1] \Leftarrow F[v_1] + A[n]B[v, v_1] \quad \text{if } X[n] = v. \qquad (21)$$

where $B[v, v_1]$ is defined in (7), (10), (12), or (13) and $A[n]$ is defined in (16) or (17).

Although the concept of varying the adjusting step was also adopted in [21], the details are considerably different. They applied a threshold, such that if the sum of the frequency table exceeds the threshold, the frequency table is multiplied by a scaling factor. Conversely, when using (16) and (17), the size of the adjusting step is increased for every n and hence a higher compression ratio can be achieved.

D. *Mutual-Learning Scheme*

In data compression, context modeling [8-10] is applied to classify the input into several conditions and different probability models are assigned for different contexts (i.e., different conditions). We propose the *mutual-learning scheme* to further improve the coding efficiency of context modeling.

Suppose that $F_j[v]$ and $F_q[v]$ are the frequency tables corresponding to the jth and the qth contexts, respectively. If the jth context and the qth context have high correlation, it is reasonable to think that, if $F_j[v]$ is adjusted, then $F_q[v]$ should also be adjusted.

For example, when performing lossless coding for an image, if the jth and the qth contexts corresponding to the cases where the prediction error $\in$ [15, 25] and the prediction error $\in$ [25, 35], respectively, then it is reasonable think that the probability distributions corresponding to the two contexts are highly correlated. If $F_j[v]$ is high, then $F_q[v]$ should also have a higher value.

Owning to the above concept, we can further modify the formula for frequency table adjusting as follows. Suppose that context modeling is adopted in the AAC process and the current context is j. If $X[n] = v$, analogous to (17), the frequency table corresponding to the jth context is adjusted as

$$F_j[v_1] \Leftarrow F_j[v_1] + A[n]B[v, v_1]. \qquad (22)$$

Moreover, the frequency table corresponding to the qth context is also adjusted as

$$F_q[v_1] \Leftarrow F_q[v_1] + A[n]B[v, v_1]C[j, q] \qquad (23)$$

where $C[j, q]$ is between 0 and 1. It depends on the similarity between the jth and qth contexts. If the characteristic of the jth context is near to that of the qth context, then $C[j, q]$ should be greater.

E. *Local Frequency Table*

Moreover, we can apply the techniques of a *local frequency table* to further improve the coding efficiency. That is, when encoding $X[n]$ according to the frequency table used in the AAC/ CAAC process (we call it the global frequency table), one can also apply the causal information (i.e., the information of the previous inputs $X[n-1]$. $X[n-2]$, $X[n-3]$, ……) to construct a local frequency table based on the global frequency table:

$$F_L[v] = \rho[v]F[v], \quad \rho[v] > 1 \tag{24}$$

if it is predicted that $X = v$ has a greater probability than the usual case and

$$F_L[v] = F[v] \text{ otherwise.} \tag{25}$$

If it is estimated that $P_1 = P(X[n] = v)$ in the current case and $P_2 = P(X[n] = v)$ in the usual case, then $\rho[v]$ can be set to P_1/P_2. Then, in Fig. 1, we apply the local frequency table $F_L[v]$ instead of $F[v]$ to encode the input data. After $X[n]$ is coded, only the global frequency table is adjusted. Hence, the data could be coded using a local table, while maintaining the integrity of the true frequency table.

For example, assume that we were to perform lossless image compression using CAAC to encode the prediction residue. In a digital image, the probability that $X[m, n]$ is equal to $X[m-1, n]$ or $X[m, n-1]$ is normally considerably high. One can use this fact to construct the local frequency table F_L from the global frequency table F. That is, in (24), $\rho[v]$ could be set to greater than 1 if $v = X[m-1, n] - X_e[m, n]$ or $v = X[m, n-1] - X_e[m, n]$ where $X_e[m, n]$ is the predicted value of $X[m, n]$ in the EDP process. See Section III-A.

Although several new techniques are adopted, the complexity of the proposed improved AAC remains linear to the input data size. The techniques of initializing the frequency table, increasing the adjusting step, and the local frequency table do not much affect the computation time. The time requirement for the range-adjusting scheme and mutual-learning can be reduced by table lookup, i.e., storing the values of $B[v, v_1]$ and $C[j, q]$ in (7), (10), (12), (13), and (23) in advance.

III. Applications

In Section II, we proposed five techniques to improve the coding efficiency of AAC. In Sections III, we provide five examples that apply the proposed coding algorithm for

(i) lossless image compression,
(ii) encoding DC differences in JPEG
(iii) encoding AC terms in JPEG,
(iv) bit plane coding in the JPEG2000 process, and
(v) encoding the motion vector differences (MVDs) in video compression.

All of the five examples demonstrate that the proposed techniques can achieve the best performance and are beneficial for improving the performance of data compression.

The MATLAB codes that apply the proposed coding algorithm to these five data compression problems can be downloaded from [22].

A. *Example for EDP Lossless Image Compression*

Lossless image compression is important for security and medical image processing [23-25]. After performing lossless image compression, one can perfectly reconstruct the original image without missing any information. There are some well-known lossless image compression algorithms, such as the context-based, adaptive, lossless image coding (CALIC) algorithm in [24]. In [25], Li and Orchard applied the EDP algorithm for lossless image compression. It has even superior coding efficiency. The EDP algorithm applies the linear combination of adjacent pixels to predict the current pixel value:

$$X_e[\mathbf{n}] = \sum_{k=1}^{K} c_k X[\mathbf{n} - \mathbf{d_k}] \tag{26}$$

where X is the original image, X_e is the predicted image, $\mathbf{n} = [m, n]$, and

$$\mathbf{d_k} = [i_k, j_k], \quad \text{where } i_k > 0 \quad \text{or} \quad i_k = 0 \text{ and } j_k > 0,$$
$$\sqrt{i_k^2 + j_k^2} \le L \text{ where } L \text{ is some threshold.} \tag{27}$$

The linear combination coefficients c_k are determined by the least mean square error (LMSE) method [25]. Then, AAC together with context modelling are applied to encode the residue (i.e., the prediction error):

$$E[\mathbf{n}] = X[\mathbf{n}] - X_e[\mathbf{n}]. \tag{28}$$

In the following, we describe how to use the proposed improved AAC algorithm to improve the coding efficiency of EDP. Moreover, we also describe the proposed context modeling method for EDP.

It is a pity that the explicit method to construct the context was not described clearly in [25]. They only stated that the context modeling method they used was similar to that of CALIC. We attempted several ways to construct the context and found that it would be better to apply the difference of the prediction results of EDP and gradient adjusted prediction (GAP), which is adopted by CALIC, to construct the context. First, we calculate

$$\Delta = 0.8|d_h + d_v| + 2.4\varepsilon \tag{29}$$

$$\begin{aligned}
\varepsilon = (&|X[m,n-1] - X_e[m,n-1]| + \\
&|X[m-1,n] - X_e[m-1,n]|)/2, \\
d_h = &|X[m,n-1] - X[m,n-2]| + |X[m-1,n] - X[m-1,n-1]| \\
&+ |X[m-1,n+1] - X[m-1,n]|, \\
d_v = &|X[m,n-1] - X[m-1,n-1]| + |X[m-1,n] - X[m-2,n]| \\
&+ |X[m-1,n+1] - X[m-2,n+1]|.
\end{aligned} \tag{30}$$

Then, we choose thresholds as follows:

$$t_1[s] = (t_0[s_1])^{1-s_2} (t_0[s_1+1])^{s_2} \tag{31}$$

for $s = 1, 2, \ldots, C_1 - 1$ (C_1 is some constant), $t_1(0) = 0$, and $t_1(C_1) \to \infty$, and

$$t_0[1] = 0.5, \quad t_0[b] = 2.6 \cdot 1.75^{b-1} \text{ for } b > 1, \quad s_0 = 8s/C_1,$$
$$s_1 = 1 + \text{floor}(s_0), \quad s_2 = s_0 - s_1 + 1. \tag{32}$$

If $t_1(s_a - 1) \le \Delta < t_1(s_a)$ where s_a is an integer, then we set the context related to the texture as:

$$T_1 = s_a. \tag{33}$$

Note that $1 \le T_1 \le C_1$. Then, suppose that X_e and $X_{e,1}$ are the prediction results of EDP and GAP, respectively. We set $t_2[0] = -\infty$, $t_2[1] = -7$, $t_2[2] = -3$, $t_2[3] = -1$, $t_2[4] = 0$, $t_2[5] = 1$, $t_2[6] = 2$, $t_2[7] = 4$, $t_2[8] = 8$, and $t_2[9] = \infty$. If $t_2[\rho-1] \le X_e - X_{e,1} < t_2[\rho]$, then we set

$$T_2 = \rho. \tag{34}$$

Then, the residue $E[m, n] = X[m, n] - X_e[m, n]$ is encoded by the C^{th} context where

$$C = 9(T_1 - 1) + T_2. \tag{35}$$

If we choose $C_1 = 14$, then because $1 \le T_1 \le 14$ and $1 \le T_2 \le 9$, there are 126 contexts. Compared to the context modeling method of the standard CALIC process, which uses 576 contexts, the proposed algorithm uses a considerably reduced number of contexts. However, higher coding efficiency can be achieved.

Next, we discuss the way to use the proposed coding algorithm to encode the residue $E[\mathbf{n}]$ in (28). We initialize the frequency table using different Gaussian models for different contexts:

$$F_c[v] = Max\left\{A\exp\left[-(v-m_0)^2/2\sigma^2\right], 0.1\right\}, \tag{36}$$

$$m_0 = (1.98 + 3.3\cdot 1.3^{0.05T_1})\frac{T_2-5}{9}, \quad \sigma = 2 + 2\cdot 1.4^{0.11T_1},$$

$$A = 1000 / \sum_v \exp\left[-(v-m_0)^2/2\sigma^2\right]. \tag{37}$$

Then, we apply the *range-adjusting* scheme proposed in Section II-B to adjust the frequency table. If the residue is v, then in the original CAAC algorithm, only the value of $F_c[v]$ is adjusted. However, we think that the probability that the residue is τ has very high correlation with the probability that the residue is v if v is near to τ. Thus, in addition to $F_c[v]$, $F_c[\tau]$ should also be adjusted, especially when $|v - \tau|$ is small. Therefore, we apply the following method to adjust the frequency table if the residue is v:

$$F_c[v] \Leftarrow F_c[v] + A\cdot\delta(v) \quad \text{when } \tau = 0, \tag{38}$$

$$F_c[v] \Leftarrow F_c[v] + A\cdot N_1^{-1}\exp\left(-\frac{|v-\tau|}{|\tau|/8}\right) \quad \text{when } \tau \ne 0$$

$$\text{where} \quad N_1 = \sum_{v=-255}^{255}\exp\left(-\frac{|v-\tau|}{|\tau|/8}\right) \tag{39}$$

$F_c[v]$ will be convergent to the true distribution of the input data more quickly than the original method in (2).

Moreover, we also apply the technique of the *increasingly adjusting step size* proposed in Section II-C. We adopt (17) and choose $d = 500$ and $c = 1.02$.

Furthermore, we also apply the *mutual-learning scheme* proposed in Section II-D to adjust the frequency table. As in (35), we use 126 contexts in the EDP process. However, for the contexts with similar values of T_1 or T_2, the probability models are similar. Therefore, if the current context is c, in addition to adjusting the value to $F_c[v]$ by (39), we suggest that the value of $F_d[v]$ where $d = c \pm 9$ (i.e., the case where the values of T_1 are differed by 1 and the values of T_2 are the same) or ceil(d/9) = ceil(c/9) (i.e., the case where the values of T_1 are the same) should also be adjusted. That is,

TABLE I
PERFORMANCES OF EDP LOSSLESS IMAGE COMPRESSION WHEN USING THE ORIGINAL CAAC ALGORITHM, THE ALGORITHMS IN [26] AND [28], AND THE PROPOSED IMPROVED AAC ALGORITHMS (IN TERMS OF BITS PER PIXEL (BPP)).

Images	EDP (by the original CAAC)	EDP (by the method in [26])	EDP (by the method in [28])	EDP (by the proposed algorithm)
Lena	4.0044	4.0350	4.0036	3.9606
Baboon	5.8189	5.8172	5.7853	5.7416
Peppers	4.3169	4.3581	4.3245	4.2678
Airplane	3.686	3.7351	3.6758	3.6149
Tiffany	3.865	3.9295	3.8870	3.8186
Splash	3.5343	3.5947	3.5399	3.4867
Sailboat	4.8838	4.8892	4.8394	4.8043
House	3.9686	3.9690	3.9067	3.8747
Average	**4.2597**	**4.2910**	**4.2453**	**4.1961**

TABLE II
PERFORMANCES OF EDP LOSSLESS IMAGE COMPRESSION WHEN USING ONLY PARTS OF THE PROPOSED TECHNIQUES (IN TERMS OF AVERAGE BPP).

	None	(i)	(i)+(ii)	(i)+(ii)+(iii)	(i)+(ii)+(iii)+(iv)	ALL
Average	4.2597	4.2506	4.2214	4.2162	4.1997	4.1961

* (i): initialization of the frequency table; (ii): range-adjusting scheme; (iii): increasingly adjusting step; (iv) mutual-learning scheme; (v): local frequency table

$$F_{c\pm 9}[v] \Leftarrow F_{c\pm 9}[v] + 0.6\frac{A}{N_1}\exp\left(-\frac{|v-\tau|}{|\tau|/8}\right), \tag{40}$$

$$F(c_1, v) \Leftarrow F(c_1, v) + 0.6\rho_{c_1,c}\frac{A}{N_1}\exp\left(-\frac{|v-\tau|}{|\tau|/8}\right)$$

$$\text{if } ceil(c_1/9) = ceil(c/9), \tag{41}$$

$$\text{where} \quad \rho_{c_1,c} = 0.2^{1-\frac{\zeta-0.05}{9}}\,0.06^{\frac{\zeta-0.05}{9}},$$

$$\zeta = \left|\text{mod}(c_1-1, 9) - \text{mod}(c-1, 9)\right|.$$

It frequently happens that the value of a pixel is equal to those of the adjacent pixels (i.e., $X[m, n] = X[m-1, n]$ or $X[m, n-1]$). Therefore, as in Section II-E, we apply a *local frequency table* to improve coding efficiency. In (24), we set $\rho = 1.05$ to increase the probability that the prediction residue is equal to $X[m-1, n] - X_e[m, n]$ or $X[m, n-1] - X_e[m, n]$ (i.e., $X[m, n] = X[m-1, n]$ or $X[m, n-1]$).

In Table I, several simulations are performed to test the coding efficiency of the proposed modified AAC algorithm for EDP.

We use eight well-known 512×512 images (Lena, Baboon, Peppers, Airplane, Tiffany, Splash, Sailboat, and House) as the test data. They are the gray level counterparts of the color-images from the USC-SIPI database. We illustrate the coding performances of the EDP lossless image compression method when using the original AAC and the proposed improved AAC algorithms in terms of bit per pixel (bpp). Moreover, the results when only parts of the proposed techniques are adopted are indicated in Table II. The simulation results in Table I show that, when using the proposed improved AAC algorithm, the coding efficiency of EDP is considerably superior to the case where the original CAAC algorithm is adopted. Recently, Masmoudi *et al.*

used nonparametric, semi- parametric, or parametric-defined initial frequency tables for lossless image compression [26-28]. They applied neighboring coding blocks to create the mixed initial probability model and improved the efficiency of entropy coding. In Table I, we also display the results when using the algorithms in [26] and [28] for EDP-based lossless image compression. The simulation results in Table I demonstrate that the proposed algorithm also outperforms the advanced entropy coding algorithms in [26] and [28].

B. Encoding DC and AC Terms in JPEG

In the standard JPEG process, an image is separated into 8×8 blocks. Each block is transformed by the discrete cosine transform (DCT). Then, the Huffman code is used for encoding the DC differences and the run-length coded AC coefficients. In this section, we provide examples that apply the proposed techniques in Section II to improve the coding efficiencies of DC differences and AC terms in the JPEG process.

1) DC Differences

When using the proposed improved AAC instead of the Huffman code to encode the DC differences in the JPEG process, first, we initialize the frequency table as

$$F[v] = \max\left(a_1 \exp(-0.2v), 1\right)$$
$$\text{where } a_1 = 1000 / \sum_{v=0}^{128} \exp(-0.2v) \qquad (42)$$

and v is the absolute value of DC differences. In the JPEG standard, $0 \le v \le 128$. Then, we apply the *range-adjusting scheme* of the normalized exponential form in (13) and the function $B[v, v_1]$ is chosen as:

$$B[v, v_1] = c_2 \exp\left[-7(1 - \frac{v_1}{v})^2\right] \text{ if } v \neq 0, \quad B[0, v_1] = \delta[v_1] \qquad (43)$$

where c_2 is a constant to force the sum of $B[v, v_1]$ to be 1. Then, the *increasingly adjusting step* of the geometric form is applied:

$$A[n] = 80 \cdot 1.02^n. \qquad (44)$$

Furthermore, we apply the *local frequency table*. Because the quantized DC value is in the range [0, 128], if the previous DC value is d, one can conclude that the absolute value of the DC difference should be no larger than max(d, 128–d). Therefore, one can construct the local frequency table $F_C(v)$ and set $F_C(v) = 0$ if $v > \max(d, 128-d)$. Moreover, because in the case where

$$\min(d, 128-d) < v \le \max(d, 128-d), \qquad (45)$$

the sign of the DC difference can only be positive (or negative), one can reduce the value of $F_c[v]$ in this case.

In Table III, we display the results using the proposed improved AAC to encode the DC differences of the eight color images. The test images used are 512×512 color-images from the USC-SIPI database. We also compare the result with the case where the Huffman code, SAC, and original AAC are applied. For the case of original AAC, we display two results. For the first result, the grouping method (i.e., the DC difference values of $\pm 2^k$, $\pm(2^k + 1)$, $\pm(2^k + 2)$, …, $\pm(2^{k+1} - 1)$ are grouped into a class) is not used. For the second method, the grouping method is applied.

TABLE III
USING THE PROPOSED IMPROVED ADAPTIVE ARITHMETIC CODE AND OTHER CODING METHODS TO ENCODE DC DIFFERENCES (INCLUDING Y, CB, AND CR PARTS) IN THE JPEG PROCESS (IN TERMS OF BPP).

Color Images	Huffman Code	SAC	Original AAC (I)	Original AAC (II)	Proposed Algorithm
Lena	0.115	0.111	0.115	0.111	0.109
Baboon	0.126	0.119	0.122	0.118	0.116
Peppers	0.128	0.123	0.127	0.123	0.121
Airplane	0.100	0.095	0.100	0.095	0.090
Tiffany	0.111	0.106	0.110	0.106	0.103
Splash	0.105	0.100	0.103	0.100	0.096
Sailboat	0.121	0.118	0.122	0.118	0.115
House	0.104	0.102	0.106	0.101	0.095
Average	**0.114**	**0.109**	**0.113**	**0.109**	**0.105**

TABLE IV
AVERAGE BPP OF DC DIFFERENCE WHEN ONLY PARTS OF THE PROPOSED TECHNIQUES ARE APPLIED.

	None	(i)	(i)+(ii)	(i)+(ii)+(iii)	ALL
Average	0.109	0.108	0.107	0.106	0.105

* (i): initialization of the frequency table; (ii): range-adjusting scheme; (iii): increasingly adjusting step; (v): local frequency table

The simulation results in Table III confirm that the proposed algorithm outperforms all of the other entropy coding algorithms for encoding the DC difference. In Table IV, we display the results when only parts of the five proposed techniques in Section II are applied to encode the DC differences.

2) AC Coefficients

In the standard JPEG compression procedure, AC coefficients are grouped into ten categories and the group number, together with the zero run length (ZRL), are encoded by the Huffman code. We now discuss a method to use the proposed improved AAC to encode the AC coefficients. For a color image, there are three channels: Y, Cb, and Cr. Each channel requires two frequency tables: one is for the run-length/group pair and the other one is for the appended index. In sum, there are six frequency tables.

During the *table initialization* stage, we initialize the frequency table value of (0/0) to 20 for (Y, Cb, Cr). For the remaining run-length/group pairs (denoted by (i/j)), the frequency table values are initialized as:

$$F[i, j] = f / 2^{i+j} \qquad (46)$$

where f = 50 for Y and f = 5 for Cb and Cr. The frequency table for the appended index is initialized as:

$$F[v] = 10 \exp\left(-(|v| - |c|)^2 / 2\sigma^2\right) \qquad (47)$$

where v is the AC coefficients, c is the minimal absolute value in the group (e.g., because the AC coefficients in the third group are –7, –6, –5, –4, 4, 5, 6, and 7, the value of c for the third group is 4), and σ is chosen to be 1.2 in our simulations. In the *range-adjusting* process, we increase the frequency count of similar symbols together according to a weighted factor:

$$F[v] = F[v] + A[n] \cdot f(dist(v, s)) \qquad (48)$$

TABLE V
USING THE PROPOSED IMPROVED ADAPTIVE ARITHMETIC CODE AND OTHER CODING METHODS TO ENCODE AC COEFFICIENTS IN THE JPEG PROCESS (IN TERMS OF BPP).

Color Images	Huffman Code	SAC	Original AAC	Proposed Algorithm
Lena	0.638	0.596	0.582	0.574
Baboon	1.409	1.330	1.316	1.303
Peppers	0.659	0.597	0.580	0.572
Airplane	0.664	0.634	0.620	0.608
Tiffany	0.576	0.528	0.512	0.504
Splash	0.488	0.426	0.411	0.399
Sailboat	0.907	0.843	0.828	0.819
House	0.860	0.807	0.793	0.778
Average	**0.775**	**0.720**	**0.705**	**0.695**

TABLE VI
AVERAGE BPP WHEN ONLY A SUBSET OF PROPOSED TECHNIQUES IS APPLIED TO ENCODE JPEG AC COEFFICIENTS.

	None	(i)	(i)+(ii)+(iv)	(i)+(ii)+(iii)+(iv)	ALL
Average	0.705	0.701	0.701	0.698	0.695

* (i): initialization of the frequency table; (ii): range-adjusting scheme; (iii): increasingly adjusting step; (iv): mutual-learning scheme; (v): local frequency table

where $s = (run_s, grp_s)$ indicating the values of the run-length and the group of the n^{th} input, $v = (run_v, grp_v)$ denoting the symbol whose frequency table value is to be adjusted, and

$$f\left(dist\right) = 1/\,dist^{p}. \qquad (49)$$

The *range-adjusting scheme* is applied for

(1) symbols sharing the same group as the input;

(2) symbols sharing the same run-length as the input;

(3) encoding group indexes

The value of p in cases (1), (2), and (3) are set to 6, 9, and 3, respectively. *dist* is a function that measures the distance between two sets of run-length pairs or two group indexes. It is defined as follows:

$$dist = |run_v - run_s| + 1 \quad \text{for case (1),}$$
$$dist = |grp_v - grp_s| + 1 \quad \text{for case (2), and}$$
$$dist = \left|index_v - index_s\right| + 1 \quad \text{for case (3).} \qquad (50)$$

Moreover, we also apply the *increasingly adjusting step* in Section II-C and set $A[n]$ in (48) as $A[n] = 1.0006^n$. Because there are three channels, the *mutual-learning scheme* introduced in Section II-D can also be applied in this case. When encoding the Cb (or Cr) channel, the frequency tables corresponding to the Cr (or Cb) channel can also be adjusted at the same time, however, with a smaller adjusting step.

Finally, *the local frequency table* method is adopted. Zigzag scanned DCT coefficients have a tendency to have trailing zeros. After all non-zero AC coefficients in the block have been processed, the end-of-block (EOB) symbol is encoded. It informs the decoder that the remaining of AC coefficients are zeros. Hence, we can expect that the EOB symbol has an increasing tendency to appear as we encode more and more AC coefficients in a given block. Therefore, in (24), we assign the multiplier for the EOB as

$$\rho[\text{EOB}] = num/16 + 1 \qquad (51)$$

where *num* stands for the number of AC coefficients encoded in that block and ρ is applied to the frequency count of the EOB symbol. Furthermore, in a stream of the ZRL code of AC coefficients, if (15, 0) appears, it is certain that it will not be followed by an EOB symbol. This phenomenon induced by the coding structure can be compensated by another *local frequency table* to temporarily suppress the frequency count of the EOB symbol. In this case, ρ[EOB] is set to zero.

In Table V, we display the results of applying the Huffman code, SAC, original AAC, and the proposed modified AAC algorithm to encode AC terms and the zero run lengths in the JPEG process for the eight test images. For the SAC case, the number of bits that are used for encoding the probability distribution of the data is included. It can be calculated from

$$b_{table} = B\sum_{k=1}^{N} S_k \qquad (52)$$

where N is the number of frequency tables, S_k is the number of possible values of the input data, and B is the number of bits to represent the frequency count of each symbol. Table V indicates that the proposed improved AAC algorithm demonstrates higher coding efficiency than the other entropy coding methods.

In Table VI, we provide a detailed analysis of the improvements made by each of the proposed techniques introduced in Section II.

3) DC+AC Combined

In Table VII, the results in Table III (DC differences) and Table V (AC terms) are combined to show the total number of bits when using the improved AAC algorithm in the JPEG process. One can see that, with the proposed entropy coding algorithm, the coding efficiency of JPEG can be much improved.

TABLE VII
USING THE PROPOSED IMPROVED ADAPTIVE ARITHMETIC CODE AND OTHER ENTROPY CODING METHODS TO ENCODE DCT COEFFICIENTS IN THE JPEG PROCESS (TEST IMAGES ARE 512 ×512 COLOR IMAGES) (IN TERMS OF BPP).

Color Images	Huffman Code	SAC	AAC(1)	AAC(2)	Proposed Algorithm
Lena	0.753	0.708	0.698	0.693	0.683
Baboon	1.536	1.448	1.438	1.434	1.419
Peppers	0.786	0.721	0.707	0.704	0.692
Airplane	0.764	0.729	0.719	0.715	0.698
Tiffany	0.688	0.634	0.622	0.618	0.607
Splash	0.592	0.526	0.514	0.510	0.495
Sailboat	1.029	0.962	0.951	0.946	0.934
House	0.964	0.909	0.899	0.894	0.873
Average	**0.889**	**0.830**	**0.818**	**0.814**	**0.800**

C. Example for JPEG2000

The standard JPEG 2000 process [18-20, 29] applies the methods of the 9/7 discrete wavelet transform, quantization, bit plane conversion, the embedded block coding with optimized truncation (EBCOT) scheme, and the MQ coder for image compression.

TABLE VIII
USING CONVENTIONAL METHODS AND THE PROPOSED METHOD IN THE JPEG2000 PROCESS (TEST IMAGES ARE 512×512 COLOR IMAGES) (IN TERMS OF BPP).

Color Images	Static Arithmetic	CAAC	MQ coder	Proposed Improved AAC
Lena	2.198	2.200	2.142	2.094
Baboon	6.467	6.466	6.409	6.273
Peppers	2.439	2.440	2.300	2.252
Airplane	1.730	1.729	1.623	1.592
Tiffany	2.437	2.436	2.369	2.323
Splash	1.408	1.407	1.317	1.297
Sailboat	4.656	4.655	4.551	4.460
House	2.666	2.665	2.501	2.455
Average	**3.000**	**3.000**	**2.902**	**2.843**

TABLE IX
APPLYING ONLY A SUBSET OF THE PROPOSED TECHNIQUES TO ENCODE (CX, D) PAIRS (IN TERMS OF BPP).

	None	(i)	(i)+(ii)+(iii)	ALL
Average	3.000	2.938	2.848	2.843

* (i): initialization of the frequency table; (ii): range-adjusting scheme; (iii): increasingly adjusting step; (iv): mutual-learning scheme

One of the main reasons that JPEG2000 has superior coding efficiency compared to JPEG is that the entropy coding method of the MQ coder is applied. The MQ coder is suitable to compress binary data and its coding efficiency is even better than that of Huffman code, SAC, and conventional CAAC.

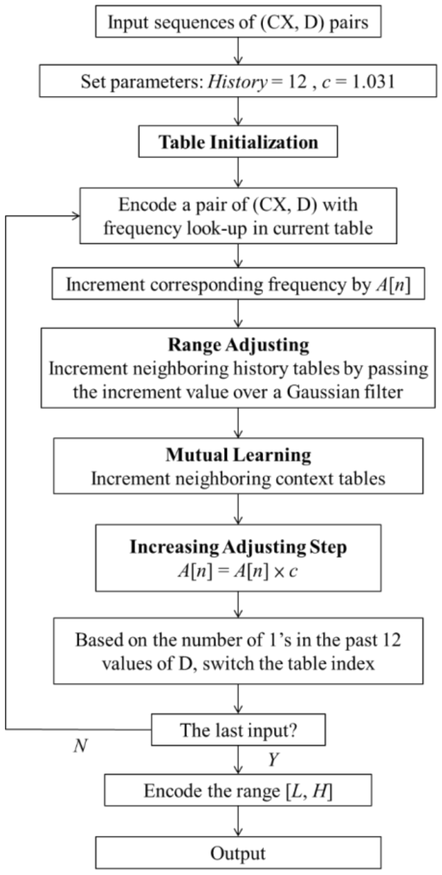
We now discuss how to apply the proposed improved AAC algorithm to improve the coding efficiency of JPEG2000 and provide simulation examples. To ensure a fair comparison, except for replacing the MQ coder by the proposed improved AAC algorithm, we *do not modify other parts of the JPEG2000 process.*

The input image is first processed by the wavelet transform and the EBCOT scheme, which outputs a sequence of (Context (CX), Decision Bit (D)) pairs, with 19 possible CX symbols [29]. The (CX, D) pairs are then coded through the flow chart as indicated in Fig. 3. In Fig. 3, *History* represents the amount of the previous data that we have checked and c is the factor by which we multiply in the adjusting step, as in (17).

In the *table initialization* process, the frequency table is initialized linearly according to its History index H_k, which means that in the past data there are k ones. The frequency table value for H_0 is initialized to 13, for H_1 is initialized to 12, …, and for H_{12} is initialized to 1. The reason behind this setup is that, if there are more ones in the past, then the probability that the new input is zero is less.

In the *range-adjusting* stage, the frequency table is incremented by a function that has an exponential form as in (12). We choose $\alpha = 2$ to force the increment to be similar to a Gaussian function.

The *increasingly adjusting step* of the geometric form in (17) is adopted in this case with c = 1.031. After $A[n]$ is modified, the coder switches the context based on the number of ones of the previous coded data. The process continues until all (CX, D) pairs have been encoded.



Input sequences of (CX, D) pairs
Set parameters: $History = 12$, $c = 1.031$
Table Initialization
Encode a pair of (CX, D) with frequency look-up in current table
Increment corresponding frequency by $A[n]$
Range Adjusting
Increment neighboring history tables by passing the increment value over a Gaussian filter
Mutual Learning
Increment neighboring context tables
Increasing Adjusting Step
$A[n] = A[n] \times c$
Based on the number of 1's in the past 12 values of D, switch the table index
The last input?
N
Y
Encode the range [L, H]
Output

Fig. 3. Flowchart of the proposed improved AAC algorithm to encode (CX, D) pairs in the JPEG2000 process.

The *mutual-learning* method is applied by allowing neighboring contexts that belong to different CX symbols to have their frequency tables adjusted accordingly. There are 19 contexts for JPEG2000 [29]. We apply the mutual-learning scheme for the contexts related to zero coding (i.e., CX = 0, 1, …, 8). We do not apply it to the remaining contexts.

In Table VIII, the performances of applying conventional methods (the MQ encoder, SAC, and original CAAC) and the proposed improved AAC algorithm to encode the (CX, D) pairs in the JPEG2000 process are listed. The same set of color images as in Section III-B is tested. Note that, for SAC, the number of bits required for encoding the probabilities (4 bytes for each context, with a total of 76 bytes) is included.

The results in Table VIII demonstrate that the proposed algorithm has improved performance compared to the state-of-the-art algorithms (including the MQ coder) for encoding the bit planes in the JPEG2000 process.

In Table IX we provide an analysis of how each of the proposed techniques improves the coding efficiency. Note that, except for the local frequency table, which is not applied in this case, other techniques are beneficial for improving the coding efficiency of JPEG2000. The proposed algorithm has a superior performance to the MQ coder even if only three of the proposed techniques (the initial frequency table, the range-adjusting scheme, and the increasingly adjusting step size) are applied.

TABLE X
USING DIFFERENT CODING METHODS FOR BIT PLANE CODING IN JPEG2000. THE QUANTIZATION LEVEL IS ADJUSTED PROPERLY TO FIX THE MQ CODER TO 1 BPP.

Images	Static Arithmetic	Adapt Arithmetic	MQ coder	Proposed Improved AAC
Lena	1.047	1.048	1.000	0.986
Baboon	1.051	1.051	1.000	0.986
Peppers	1.048	1.049	1.000	0.984
Airplane	1.076	1.077	1.000	0.989
Tiffany	1.057	1.057	1.000	0.983
Splash	1.077	1.078	1.000	0.981
Sailboat	1.050	1.051	1.000	0.985
House	1.082	1.083	1.000	0.981
Average	**1.061**	**1.062**	**1.000**	**0.984**

TABLE XI
USING DIFFERENT CODING METHODS FOR BIT PLANE CODING IN JPEG2000. THE QUANTIZATION LEVEL IS ADJUSTED PROPERLY TO FIX THE MQ CODER TO 0.5 BPP.

Images	Static Arithmetic	Adapt Arithmetic	MQ coder	Proposed Improved AAC
Lena	0.532	0.530	0.500	0.490
Baboon	0.539	0.537	0.500	0.489
Peppers	0.534	0.532	0.500	0.490
Airplane	0.557	0.555	0.500	0.492
Tiffany	0.540	0.538	0.500	0.491
Splash	0.561	0.559	0.500	0.492
Sailboat	0.533	0.531	0.500	0.490
House	0.553	0.551	0.500	0.491
Average	**0.543**	**0.542**	**0.500**	**0.491**

In Tables X and XI, we perform two additional simulations. We adjust the quantization table for each image iteratively to make the bit rates of the MQ coder equal to 0.5bpp or 1bpp in all cases. The results in Tables X and XI indicate that when encoding the bit plane in JPEG2000, the proposed algorithm outperforms other entropy coding methods, including the MQ coder, even in the low bit-rate case.

D. Example for Motion Vector Difference Encoding in Video Compression

In video compression, motion vectors (MVs) are the side information in the compression procedure. Typical motion vector coding methods utilize prediction to generate motion vector differences (MVDs). Then the MVDs are encoded by entropy coding techniques.

In this subsection, we first implement an H.264/AVC motion vector predictor to acquire the MVDs. It adopts the motion vectors in the left, upper left (UL), upper (U) and upper right (UR) blocks to perform motion vector prediction (MVP) by (53). Each motion vector is predicted by at most three adjacent MVs. Then, MVD can be acquired by (54), which is the residue of subtracting MVP from the MV in the current block (MV$_C$).

$$MVP = \begin{cases} 0, & \text{no reference block} \\ \text{median}\left(MV_L, MV_U, MV_{UR}\right), & \text{if UR exists} \\ \text{median}\left(MV_L, MV_U, MV_{UL}\right), & \text{otherwise} \end{cases} \quad (53)$$

TABLE XII
USING CONVENTIONAL METHODS AND THE PROPOSED IMPROVED AAC FOR MOTION VECTOR DIFFERENCE ENCODING (IN TERMS OF BITS PER MVD).

Video	SAC	AAC	Proposed Improved AAC
Akiyo_QCIF	1.082	1.081	1.090
Carphone_QCIF	5.910	5.897	5.531
City_QCIF	7.630	7.622	6.894
Coastguard_QCIF	5.531	5.523	5.104
Grandma_QCIF	2.048	2.071	2.013
Hall_QCIF	1.856	1.868	1.900
Ice_QCIF	6.990	6.957	6.706
Suzie_QCIF	6.009	5.992	5.638
Foreman_CIF	8.128	8.115	7.703
MissAmerica_CIF	5.773	5.764	5.221
Mobile_CIF	5.743	5.739	5.167
Stefan_CIF	6.601	6.589	5.533
Waterfall_CIF	4.044	4.044	3.077
Average	5.180	5.174	4.737

$$MVD = MV_C - MVP\,. \quad (54)$$

Then, the proposed techniques in Section II, including table initialization and the local frequency table, are applied to encoding the MVD in (54), i.e., the residue of MV$_C$.

During the *table initialization* process, the frequency table is initialized as:

$$F[v] = \max\left[a \times e^{-\sigma(v-1)}, \kappa\right]$$

where a = 60, σ = 7.0, and κ = 1. This setup is based on the assumption that the distribution of MVDs is centered at zero, which is common for normal videos. Moreover, the proposed *local frequency table* technique is also adopted here,

$$F_L[v] = \alpha_v \cdot F[v]$$

where F_L is the temporary *local frequency table* for encoding of the current symbol and α_v is a multiplier depending the previous MVD. To setup α_v, we implement a preliminary analysis on the transactions of MVD symbols: MVD_x and MVD_y. We found that MVDs are located primarily in the first five values, which are MVD_x, $MVD_y \in \{0, 0.25, -0.25, 0.5, -0.5\}$. Therefore, we set α by following the transition tendency of these five symbols. By observing the transition of MVDs in three types of video sequences: Akiyo (small motion), Foreman (median motion) and Stefan (large motion), we design two parameters as in (55):

$$C_x = \begin{bmatrix} 6 & 2 & 2 & 1 & 1 \\ 1 & 4 & 1 & 1 & 1 \\ 1 & 1 & 3 & 1 & 1 \\ 1 & 1 & 1 & 4 & 1 \\ 1 & 1 & 1 & 1 & 4 \end{bmatrix}, \quad C_y = \begin{bmatrix} 4 & 1 & 1 & 1 & 1 \\ 1 & 8 & 1 & 2 & 1 \\ 1 & 2 & 4 & 1 & 1 \\ 1 & 1 & 1 & 4 & 1 \\ 1 & 1 & 1 & 1 & 4 \end{bmatrix} \quad (55)$$

where C_x and C_y are the fixed parameters for encoding MVD_x and MVD_y, respectively. In the proposed *local frequency table*, we select the value α based on the previous symbol and its corresponding column vector. For example, if we are encoding MVD_x and the previous value is the third symbol, then we would scale the frequency table temporarily according to the third column of C_x in (55).

For the simulations in Table XII, the test data are 13 well-known videos from [30]. We extract the MVDs in the first 100 frames based on the H.264/AVC baseline [31] with $QP = 28$. The simulation results in Table XII confirm that the proposed method has the highest coding efficiency overall.

IV. CONCLUSION

In this paper, we proposed five techniques, including the range-adjusting scheme, the increasingly adjusting step, table initialization, the mutual-learning scheme, and the local frequency table, to improve the coding efficiency of AAC and CAAC. In particular, with the range-adjusting scheme, the correlation among the probabilities corresponding to different data values could be further exploited. With the mutual-learning scheme, the similarity among the probability distributions of different contexts could also be considered. Simulations confirm that with the proposed improved AAC algorithm, the compression ratio can be further improved. It has the potential to improve the coding efficiency of any type of data, including images, videos, texts, acoustic signals, and biomedical signals. The MATLAB codes for the proposed improved AAC algorithm in Section III are available from [22].

REFERENCES

[1] I. M. Pu, Fundamental Data Compression, Butterworth-Heinemann, Oxford, MA, 2006

[2] Y. V. Ivanov and C. J. Bleakley, "Survey and Pareto analysis method for coding efficiency assessment of low complexity H. 264 algorithms," in *Proc. of 10th Irish Machine Vision and Image Processing Conf*, pp. 1-8, 2006.

[3] N. Memon, "Adaptive coding of DCT coefficients by Golomb-Rice codes," in *IEEE International Conference on Image Processing*, pp. 516-520, 1998.

[4] J. J. Ding, H. H. Chen, and W. Y. Wei, "Adaptive Golomb code for joint geometrically distributed data and its application in image coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, issue 4, pp. 661-670, Apr. 2013.

[5] K. Sayood, *Introduction to Data Compression*, Elsevier, Boston, 2006.

[6] P. G. Howard and J. S. Vitter, "Analysis of arithmetic coding for data compression," *Information Processing & Management*, vol. 28, pp. 749-763, Dec. 1992.

[7] N. Kuroki, T. Manabe, and M. Numa, "Adaptive arithmetic coding for image prediction errors," *IEEE International Symposium on Circuits and Systems*, vol. 3, pp. 961-964, 2004.

[8] 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, pp. 620-636, July 2003.

[9] J. Zhang and G. Liu, "A novel lossless compression for hyperspectral images by context-based adaptive classified arithmetic coding in wavelet domain," *IEEE Geosci. Remote Sens. Lett.*, vol. 4, pp. 461-465, 2007.

[10] J. Wang, X. Ji, S. Zhao, X. Xie, and J. Kuang, "Context-based adaptive arithmetic coding in time and frequency domain for the lossless compression of audio coding parameters at variable rate," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 1, pp. 1-13, 2013.

[11] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consum. Electron.*, vol. 38, pp. 18-34, 1992.

[12] K. Sayood, *Lossless Compression Handbook*. New York: Academic, 2002

[13] Y. Zhang and D. Adjeroh, "Prediction by partial approximate matching for lossless image compression," *IEEE Trans. Image Processing*, vol. 17, issue 6, pp. 924-935, 2008.

[14] G. A. Triantafyllidis and M. G. Strintzis, "A context based adaptive arithmetic coding technique for lossless image compression," *IEEE Signal Processing Lett.*, vol. 6, pp. 168–170, July 1999.

[15] B. Aiazzi, L. Alparone, and S. Baronti, "Context modeling for near-lossless image coding," *IEEE Signal Processing Lett.*, vol. 9, issue 3, pp.77–80, Mar. 2002.

[16] I. Matsuda, N. Ozaki, Y. Umezu and S. Itoh, "Lossless coding using variable block-size adaptive prediction optimized for each image," *Proc. of 13th European Signal Processing Conf.*, pp. 1-4, 2005.

[17] X. Wu and N. D. Memon, "Context-based, adaptive, lossless image coding," *IEEE Trans. Commun.*, vol. 45, pp. 437–444, Apr. 1997.

[18] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG2000 still image coding system: an overview," *IEEE Trans. Consum. Electron.*, vol. 46, issue 4, pp. 1103-1127, 2000.

[19] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Trans. Image Processing*, vol. 9, no. 7, pp.1158 -1170, 2001.

[20] H. Yang, M. Long, and H. M. Tai, "Region-of-interest image coding based on EBCOT," in *Vision, Image and Signal Processing, IEE Proceedings*, vol. 152, issue 5, pp. 590-596, Oct. 2005.

[21] P. G. Howard and J. S. Vitter, "Practical implementations of arithmetic coding," in *Image and Text Compression*, J. A. Storer, Ed. Boston, MA: Kluwer, pp. 85–112, 1992.

[22] Our codes are available in http://djj.ee.ntu.edu.tw/IAAC_distributed.zip.

[23] J. Kim and C. M. Kyung, "A lossless embedded compression using significant bit truncation for HD video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, issue 6, pp. 848-860, 2010.

[24] X. Wu and N. D. Memon, "Context-based, adaptive, lossless image coding," *IEEE Trans. Commun.*, vol. 45, pp. 437–444, Apr. 1997.

[25] X. Li and M. Orchard. "Edge-directed prediction for lossless compression of natural images," *IEEE Trans. Image Processing*, vol. 10, pp. 813-817, 2001.

[26] A. Masmoudi and A. Masmoudi, "A new arithmetic coding model for a block-based lossless image compression based on exploiting inter-block correlation," *Signal Image Video Process.*, vol. 9, issue 5, pp. 1021–1027, 2013.

[27] A. Masmoudi, W. Puech, and A. Masmoudi, "An improved lossless image compression based arithmetic coding using mixture of non-parametric distributions," *Multimedia Tools and Applications*, pp.1–15, 2014

[28] A. Masmoudi, A. Masmoudi, and W. Puech. "A new semiparametric finite mixture model-based adaptive arithmetic coding for lossless image compression," *Circuits, Systems, and Signal Processing*, pp.1-24, July 2015.

[29] D. S. Taubman and M. W. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards and Practice*, Kluwer Academic Publishers, Boston, 2012.

[30] Xiph.org Video Test Media [online] https://media.xiph.org/video/derf/.

[31] A. A. Muhit, M. R. Pickering, M. R. Frater, and J. F. Arnold, "Video coding using elastic motion model and larger blocks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, issue 5, pp. 661-672, 2010.

Jian-Jiun Ding was born in 1973 in Taiwan. He received the B.S. degree in 1995, the M.S. degree in 1997, and the Ph.D. degree in 2001, all in electrical engineering from the National Taiwan University (NTU), Taipei, Taiwan. During 2001 to 2006, he was a postdoctoral researcher in the Department of Electrical Engineering of NTU.
He is currently a professor with the Department of Electrical Engineering and the Graduate Institute of Communication Engineering, NTU. His current research areas include time-frequency analysis, fractional Fourier transforms, linear canonical transforms, wavelet transforms, image processing, image compression, orthogonal polynomials, fast algorithms,

integer transforms, quaternion algebra, pattern recognition, filter design, etc.

I-Hsiang Wang received the B.S degree from the Department of Electrical Engineering, National Taiwan University, Taiwan, in 2016. He will receive the M.S. degree in the Graduate Institute of Communication Engineering, National Taiwan University in 2017. His research interests include image, video, and biomedical signal compression.

Hung-Yi Chen received the B.S degree from the Department of Communication Engineering, National Chiao-Tung University, Taiwan, in 2015. He will receive the M.S. degree in the Graduate Institute of Communication Engineering, National Taiwan University in 2017. His research interests include computer vision, multimedia signal processing, and data compression.