Texture- and Multiple-Template-Based Algorithm for Lossless Compression of Error-Diffused Images

Yong–Huai Huang and Kuo–Liang Chung, *Senior Member, IEEE*

***Abstract*—Recently, several efficient context-based arithmetic coding algorithms have been developed successfully for lossless compression of error-diffused images. In this paper, we first present a novel block- and texture-based approach to train the multiple-template according to the most representative texture features. Based on the trained multiple template, we next present an efficient texture- and multiple-template-based (TM-based) algorithm for lossless compression of error-diffused images. In our proposed TM-based algorithm, the input image is divided into many blocks and for each block, the best template is adaptively selected from the multiple-template based on the texture feature of that block. Under 20 testing error-diffused images and the personal computer with Intel Celeron 2.8-GHz CPU, experimental results demonstrate that with a little encoding time degradation, 0.365 s (0.901 s) on average, the compression improvement ratio of our proposed TM-based algorithm over the joint bilevel image group (JBIG) standard [over the previous block arithmetic coding for image compression (BACIC) algorithm proposed by Reavy and Boncelet is 24%] (19.4%). Under the same condition, the compression improvement ratio of our proposed algorithm over the previous algorithm by Lee and Park is 17.6% and still only has a little encoding time degradation (0.775 s on average). In addition, the encoding time required in the previous free tree-based algorithm is 109.131 s on average while our proposed algorithm takes 0.995 s; the average compression ratio of our proposed TM-based algorithm, 1.60, is quite competitive to that of the free tree-based algorithm, 1.62.**

***Index Terms*—Arithmetic coding, context, error-diffused images, lossless compression, multiple-template, texture.**

I. Introduction

FOR facsimile transmission, bilevel image compression is a useful technique to compress the faxed document and, thus, can reduce the transmission time and the memory requirement. In the 1980s, two international standards, T.4 [1] and T.6 [2], were developed for facsimile coding. T.4 and T.6 used a modified Huffman table to encode the run length of black or white pixels. In 1996, three more efficient lossless bilevel image compression algorithms [3]–[5] were presented. Based on exclusive-or operations for two horizontally adjacent pixels, Robertson [3] presented an efficient algorithm to improve the compression performance. Swanson *et al.* [4] removed the spatial redundancy by using the wavelet transform. Gurcan *et al.* [5] utilized the multiresolution property of wavelet subbands to reduce the bit requirement for compression. Except error-diffused images, i.e., halftone images, the above five previous compression algorithms work well for bilevel images. Since there are a lot of transitions between black and white pixels in error-diffused images, neither the runlength-based coding approach nor the wavelet transform-based coding approach can work well for error-diffused images.

A series of context algorithms [6]–[8] proposed by Rissanen and Langdon is an efficient technique for data compression. According to context-based concept, several efficient algorithms for compressing error-diffused images have been presented [9]–[14]. Among these developed algorithms using arithmetic coding, the joint bilevel image group (JBIG) [9] is the most well-known. According to a ten-pixel template, the JBIG generates 1024 different contexts and each context is corresponding to a probability model to encode the pixels with the same context. Experimental results show that for error-diffused images, the JBIG has higher compression ratio when compared to the previous algorithms [1]–[4]. Based on more efficient templates, three novel algorithms [10]–[13] have been proposed to improve the compression performance of the JBIG for compressing error-diffused images. Using a 12-pixel template, Reavy and Boncelet [10] presented an efficient block-based arithmetic coding algorithm. For convenience, their proposed coding algorithm is called the block arithmetic coding for image compression (BACIC) algorithm. In [11], the same 12-pixel template was used for compressing error-diffused images. Lee and Park [12] presented a two-pass coding algorithm, each pass associated with a different template, for compressing error-diffused images. For convenience, the coding algorithm proposed by Lee and Park is called the two-pass arithmetic coding for image compression (PACIC) algorithm. Based on the PACIC algorithm, Lee and Park [13] further proposed a progressive coding algorithm to extend the capability of the PACIC algorithm to cover the progressive image transmission. In the JBIG, the BACIC algorithm, the PACIC algorithm, and the progressive coding algorithm [13], the used templates are fixed and cannot reflect texture features completely for different binary patterns in error-diffused images. Martins and Forchhammer [14] proposed a free tree-based algorithm to compress error-diffused images. For convenience, the coding algorithm proposed by Martins and Forchhammer is called the free tree arithmetic coding for image compression (FACIC) algorithm. Experimental results demonstrate that the FACIC algorithm has better compression ratio when compared to the JBIG, the BACIC algorithm, and the PACIC algorithm. However, the FACIC algorithm needs a lot of time to complete the online training phase for each input error-diffused image in order to

Manuscript received April 21, 206; revised December 15, 2006. This work was supported by the National Council of Science of Taiwan, R.O.C., under Contract NSC94-2213-E-011-041. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Bruno Carpintieri.

The authors are with the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan 10672, R.O.C. (e-mail: klchung@csie.ntust.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2007.894227

build up the near-optimal tree where each leaf is the constructed context, and, thus, the encoding time required in the FACIC algorithm is much slower than that in the JBIG, the BACIC algorithm, and the PACIC algorithm.

In this paper, we first present a novel block- and texture-based approach to train the multiple-template from the training set of error-diffused images in an offline training way. In our proposed block- and texture-based training approach, the multiple-template is constructed for the most representative texture features. Based on the constructed multiple-templates, we next present an efficient texture- and multiple-template-based (TM-based) coding algorithm, called the TMCIC algorithm, for lossless compression of error-diffused images. In our proposed TMCIC algorithm, the input error-diffused image is first divided into many blocks and for each block, the best template is adaptively selected according to the texture feature of that block. Based on 20 testing error-diffused images and the personal computer with Intel Celeron 2.8-GHz CPU, experimental results demonstrate that with a little encoding time degradation, 0.365 s (0.901 s) on average, the compression improvement ratio of our proposed TMCIC algorithm over the JBIG standard (over the BACIC algorithm) is 24% (19.4%). Further, experimental results also demonstrate that the compression improvement ratio of our proposed algorithm over the previous PACIC algorithm is 17.6% and still only has a little encoding time degradation (0.775 s on average). The encoding time required in the previous FACIC algorithm is 109.131 s on average while our proposed algorithm takes 0.995 s; the compression performance of our proposed TMCIC algorithm is quite competitive to that of the FACIC algorithm.

The remainder of this paper is organized as follows. In Section II, we survey the previous four concerned coding algorithms for lossless compression of error-diffused images. In Section III, our proposed TMCIC algorithm is presented. Section IV demonstrates the performance comparison among our proposed TMCIC algorithm and the previous four algorithms. Section V addresses some concluding remarks.

II. Past Four Works: JBIG, BACIC, PACIC, and FACIC Algorithms

For lossless compression of error-diffused images, based on the context-based arithmetic coding, the past four works, the JBIG [9], the BACIC algorithm [10], the PACIC algorithm [12], and the FACIC algorithm [14], are surveyed in this section. Since the main concern of this paper is focused on the compression and execution time performance, the survey of the progressive coding algorithm proposed by Lee and Park [13] is not considered in this section.

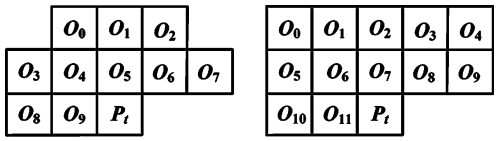

Fig. 1. Templates used in JBIG and BACIC algorithm. (a) Template T_J used in JBIG. (b) Template T_B used in BACIC algorithm.

A. JBIG and the BACIC Algorithm

The JBIG uses a ten-pixel template T_J [see Fig. 1(a)] to generate $2^{10}(=1024)$ contexts $\{C_i|0 \le i \le 1023\}$ for encoding a $W \times H$ input error-diffused image I in raster scan order. In Fig. 1(a), P_t denotes the position (x_t, y_t) of the current encoded pixel $I(P_t), 0 \le x_t \le W-1$ and $0 \le y_t \le H-1$. For the current encoded pixel $I(P_t)$, the ten offsets $\{O_i^J|0 \le i \le 9\}$ constitute the ten-pixel template T_J where the ten offsets are denoted by $O_0^J = (-1,-2), O_1^J = (0,-2), O_2^J = (1,-2), \ldots$, and $O_9^J = (-1,0)$. From the current pixel $I(P_t)$ associated with its own template T_J, the mapping address of $I(P_t)$ in the contexts $\{C_i|0 \le i \le 1023\}$ is defined by

$$k = \sum_{i=0}^{9} 2^i I\left(P_t + O_i^J\right). \tag{1}$$

Once the mapping address of $I(P_t)$, i.e., k, is computed by (1), the value of $I(P_t)$ is assigned to the context C_k and the probability of $I(P_t)$ in C_k is updated for arithmetic coding according to the Bayesian rule in (2), shown at the bottom of the page, where $N_{C_k}^t(0)(N_{C_k}^t(1))$ denotes the number of zeros (ones) assigned to the context C_k right before encoding $I(P_t)$ and the value of δ is set to 0.45 in the JBIG.

Different from the template used in the JBIG, the BACIC algorithm by Reavy and Boncelet adopts a different 12-pixel template T_B as shown in Fig. 1(b). In T_B, there are 12 offsets $\{O_i^B|0 \le i \le 11\}$ and they are denoted by $O_0^B = (-2,-2), O_1^B = (-1,-2), O_2^B = (0,-2), \ldots$, and $O_{11}^B = (-1,0)$. According to T_B, the mapping address of the current encoded pixel $I(P_t)$ in the contexts $\{C_i|0 \le i \le 4095\}$ is given by $k = \sum_{i=0}^{11} 2^i I(P_t + O_i^B)$. After calculating the mapping address, $k, I(P_t)$ is assigned to the context C_k and the probability of $I(P_t)$ in C_k is updated according to (3), shown at the bottom of the next page, where empirically δ_1 is set to 0.006 and $R_{C_k}^t(0)$ and $R_{C_k}^t(1)$ are calculated by $R_{C_k}^t(0) = (1 - I(P_t)) + \alpha R_{C_k}^{t-1}(0)$ and $R_{C_k}^t(1) = I(P_t) + \alpha R_{C_k}^{t-1}(1)$, respectively. Initially, $R_{C_k}^0(0)$ and $R_{C_k}^0(1)$ are set to 1; the forgetting factor α is set to 0.985, and it implies that the influence of the recent pixels is more important than that of the earlier pixels. Due to the adopted 12-pixel template and the probability rule, experimental results

$$p(I(P_t)|C_k) = \begin{cases} p(0|C_k) = \frac{N_{C_k}^t(0)+\delta}{N_{C_k}^t(0)+N_{C_k}^t(1)+2\delta}, & \text{if } I(P_t) = 0 \\ p(1|C_k) = 1 - p(0|C_k), & \text{otherwise} \end{cases} \tag{2}$$

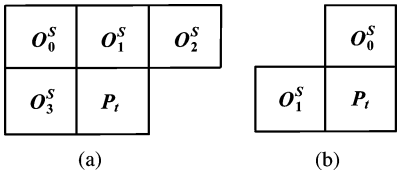

Fig. 2. Templates used in PACIC algorithm. (a) Template T_S used in first pass. (b) Template T_C used in second pass.

have demonstrated that the BACIC algorithm has better compression performance than the JBIG.

B. PACIC Algorithm

Based on the two-pass approach, Lee and Park presented a PACIC algorithm to compress the error-diffused images. According to the experimental results, the PACIC algorithm has better compression performance than those in the JBIG and the BACIC algorithm. In the PACIC algorithm, the input image is first divided into many 2×2 blocks, and for each block, the number of black pixels (or white pixels) is encoded by the first pass. Let $B(P_t)$ denote the current block at position $P_t = (x_t, y_t)$, where $0 \le x_t \le (W/2) - 1$ and $0 \le y_t \le (H/2) - 1$, and the 2×2 block $B(P_t)$ is expressed by

$$B(P_t) = \begin{bmatrix} B_{P_{B0}}(P_t) & B_{P_{B1}}(P_t) \\ B_{P_{B2}}(P_t) & B_{P_{B3}}(P_t) \end{bmatrix}$$

where P_{Bi} denotes the position (x_{Bi}, y_{Bi}) of the ith pixel in the block, $0 \le x_{Bi}, y_{Bi} \le 1$. For the current block $B(P_t)$, the number of ones in $B(P_t)$ is called the S-value and it is defined by $S(B(P_t)) = \sum_{i=0}^{3} B_{P_{Bi}}(P_t), 0 \le S(B(P_t)) \le 4$. For the current block $B(P_t)$, Fig. 2(a) shows the template T_S used in the first pass and it consists of four offsets $\{O_i^S | 0 \le i \le 3\}$ where $O_0^S = (-1,-1), O_1^S = (0,-1), O_2^S = (1,-1)$, and $O_3^S = (-1,0)$. Since the values of $S(B(P_t))$ range from 0 to 4, the number of the contexts associated with the template T_S is $5^4 (= 625)$. According to T_S, the mapping address of $S(B(P_t))$ in the contexts $\{C_i | 0 \le i \le 624\}$ is given by

$$k = \sum_{i=0}^{3} 5^i S\left(B\left(P_t + O_i^S\right)\right). \tag{4}$$

Once the value of k is obtained by (4), $S(B(P_t))$ is assigned to the context C_k and the probability of $S(B(P_t))$ in C_k is updated according to the following rule:

$$p(S(B(P_t)|C_k) = \frac{N_{C_k}^t(S(B(P_t))) + 1}{\sum_{i=0}^{4} N_{C_k}^t(i) + 5} \tag{5}$$

where $N_{C_k}^t(i)$ denotes the number of the blocks whose S-values are equal to i before encoding $S(B(P_t))$.

In the second pass, Lee and Park define a four-digit binary number to represent the so-called C-value which is used to represent the four bits in the block $B(P_t)$. Thus, the C-value of $B(P_t)$ is denoted by $C(B(P_t)) = B_{P_{B0}}(P_t) B_{P_{B1}}(P_t) B_{P_{B2}}(P_t) B_{P_{B3}}(P_t)_{(2)}$. Fig. 2(b) shows the template T_C used in the second pass, and it consists of two offsets $\{O_i^C | 0 \le i \le 1\}$ where $O_0^C = (0,-1)$ and $O_1^C = (-1,0)$. The number of contexts in the second pass is dependent on the S-value of the current block $S(B(P_t))$ and the two C-values $C(B(P_t + O_0^C))$ and $C(B(P_t + O_1^C))$. Therefore, the number of contexts is $1280 (= 5 \times 2^4 \times 2^4)$. According to the template T_C and $S(B(P_t))$, the mapping address of $C(B(P_t))$ in the contexts $\{C_i | 0 \le i \le 1279\}$ is defined by

$$k = 256 S(B(P_t)) + 16 C\left(B\left(P_t + O_0^C\right)\right) + C\left(B\left(P_t + O_1^C\right)\right). \tag{6}$$

Thus, we assign $C(B(P_t))$ to the context C_k and the probability of $C(B(P_t))$ in C_k is updated according to the following rule:

$$p(C(B(P_t))|C_k) = \frac{N_{C_k}^t(C(B(P_t))) + 1}{\sum_{i=0}^{15} N_{C_k}^t(i) + 16} \tag{7}$$

where $N_{C_k}^t(i)$ denotes the number of blocks whose C values are equal to is before encoding $C(B(P_t))$. Experimental results demonstrate that the compression performance of the PACIC algorithm is superior to that of the JBIG and is quite competitive to that of the BACIC algorithm.

C. FACIC Algorithm

In the FACIC algorithm, a free tree is constructed by using the online training and the constructed free tree is used to improve the compression performance of the JBIG. The FACIC algorithm also has better compression performance when compared to the BACIC algorithm and the PACIC algorithm. However, the time requirement of the FACIC algorithm is much more than those in the JBIG, the BACIC algorithm, and the PACIC algorithm. Fig. 3 shows an example of the free tree where the leaf nodes denote the nine contexts $\{C_i | 0 \le i \le 8\}$ and each internal node is corresponding to one of the five offsets $\{O_i^F | 0 \le i \le 4\}$ where $O_0^F = (0,-2), O_1^F = (-1,-1), \ldots,$ and $O_4^F = (-1,0)$. To select a context for the current pixel $I(P_t)$, we traverse the free tree from the root to one leaf such that the path from the root to that leaf meets the binary sequence corresponding to the neighboring pixels of $I(P_t)$. For example, if the values of $I(P_t + O_3^F), I(P_t + O_4^F)$, and $I(P_t + O_0^F)$ are 1, 0, and 0, respectively, the context C_4 is selected to the current pixel $I(P_t)$ because the traversing path is 100.

In order to obtain the N_F offsets $\{O_i^F | 0 \le i \le N_F - 1\}$ used in the free tree, a search template T_E (see Fig. 4) consisting of $(2w+1) \times h + w$ offsets $\{O_j^E | 0 \le j \le (2w+1) \times h + w - 1\}$ is used to determine the candidates for selecting these N_F offsets, i.e., $\{O_i^F | 0 \le i \le N_F - 1\} \subseteq \{O_j^E | 0 \le j \le (2w+1) \times h + w - 1\}$. The free tree corresponding to the input image is constructed by using a top-down

$$p(I(P_t)|C_k) = \begin{cases} p(1|C_k) = \frac{R_{C_k}^t(1) + \delta_1}{R_{C_k}^t(0) + R_{C_k}^t(1) + 2\delta_1}, & \text{if } I(P_t) = 1 \\ p(0|C_k) = 1 - p(1|C_k), & \text{otherwise} \end{cases} \tag{3}$$

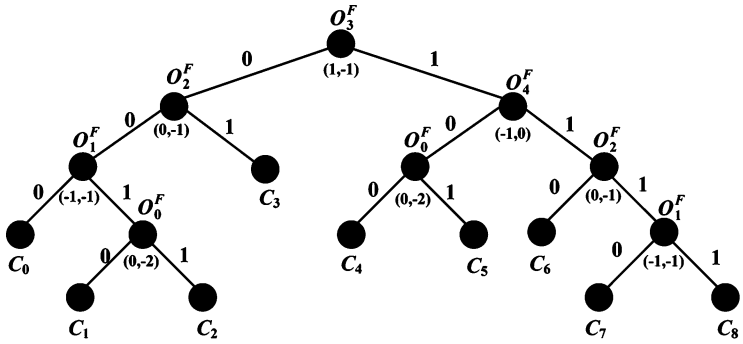

Fig. 3. Example of a free tree.

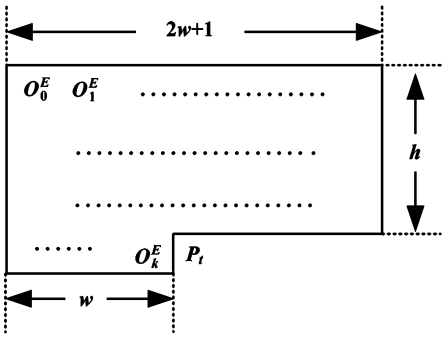

Fig. 4. Search template for free tree construction.

approach. Considering an offset O_j^E within T_E, each time a context C_l corresponding to a leaf node n_l will be split into two contexts C_{j0} and C_{j1} corresponding to two new leaf nodes called n_{j0} and n_{j1}, respectively. To maximize the compression performance, the selected offset O_j^E for splitting C_l must maximize the following compression gain:

$$\text{Gain}(C_l, C_{j0}, C_{j1}) = L_{C_l} - (L_{C_{j0}} + L_{C_{j1}} + \text{Cost}(O_j^E)) \tag{8}$$

where $L_{C_l}, L_{C_{j0}}$, and $L_{C_{j1}}$ denote the code lengths required for encoding the pixels which are assigned to C_l, C_{j0}, and C_{j1}, respectively; $\text{Cost}(O_j^E)$ denotes the cost for recording the selected offset O_j^E. The code length L_{C_l} can be defined as follows:

$$L_{C_l} = -\sum_{i=0}^{n} \log_2 p(I(P_i)|C_l) \tag{9}$$

where $I(P_i)$ denotes the ith encoded pixel in C_l; n denotes the total number of encoded pixels in C_l, and $p(I(P_i)|C_l)$ can be calculated by (2). $L_{C_{j0}}$ and $L_{C_{j1}}$ can be calculated by the same way as in (9). Once the best candidate offset O_j^E is determined, it is regarded as one of the N_F offsets $\{O_i^F|0 \le i \le N_F - 1\}$. Continue this greedy way, the free tree is growing until no context (leaf node) could be split to result in a positive gain. Based on the constructed free tree, the FACIC algorithm selects the context for each encoded pixel in the input image.

The drawback of the FACIC algorithm is that it spends much time for constructing a free tree for each input image. In Section III, based on an offline training, our proposed TMCIC algorithm is presented to construct multiple-template for different texture features. Experimental results demonstrate that the encoding time requirement of our proposed TMCIC algorithm is much less than that in the FACIC algorithm, but the compression ratio of our proposed TMCIC algorithm is quite competitive to that of the FACIC algorithm. With a little encoding time degradation, our proposed TMCIC algorithm has better compression ratio when compared to the JBIG, the BACIC algorithm, and the PACIC algorithm.

III. Our Proposed TMCIC Algorithm

As shown in Fig. 5, our proposed TMCIC algorithm is composed of the training stage, the encoding stage, and the decoding stage. Based on the block-based offline training strategy, the proposed training stage can classify the most representative texture features from a set of training images. Next, an efficient method is used to construct the template for each classified texture feature. To improve the compression performance of the context-based arithmetic coding, for each context determined by the constructed multitemplate, the probability of zeros (ones) in each context has also been estimated in training stage. In the encoding stage, the input image is divided into many blocks, and for each block, the best template will be adaptively selected from the constructed multiple-template according to the texture feature of that block. Once the best template has been determined, the address of the template is recorded and then the estimated probabilities of pixel values in each context are used to initialize the arithmetic coder. In decoding stage, for each block, the address of the best template is read and the initialization of the arithmetic coder is performed.

A. Training Stage

The block diagram for the training stage of our proposed TMCIC algorithm is shown in Fig. 6. After dividing each training image into the set of $W_B \times H_B$ training blocks, our proposed training stage first classifies the most representative texture features from the training blocks. Empirically, we set $W_B = H_B = 64$. Let $B = \{B(i)|0 \le i \le N_B - 1\}$ denote the set of N_B training blocks where $B(i)$ denotes the ith training block in B. To analyze the texture feature of $B(i)$, we apply four-level CDF 9/7 wavelet transform [15] to decompose $B(i)$ into 13 subbands $\{W_{B(i)}^j|0 \le j \le 12\}$ where $W_{B(i)}^0$ denotes the low-frequency subband and $\{W_{B(i)}^j|1 \le j \le 12\}$ denote 12 high-frequency subbands. Let $F_{B(i)} = \{F_{B(i)}(j)|0 \le j \le 12\}$

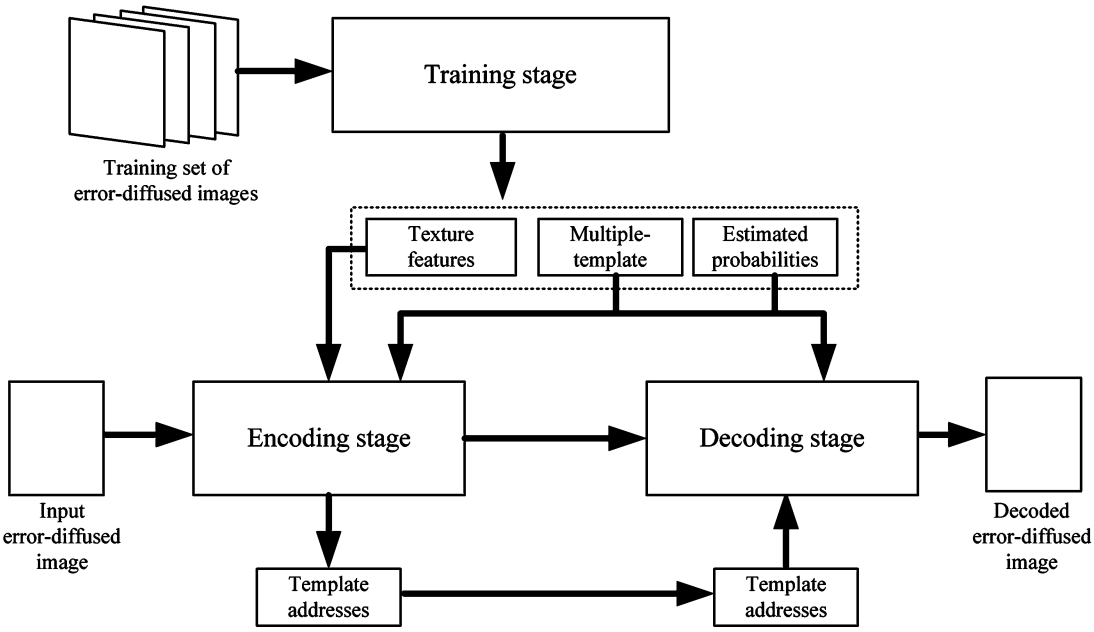

Fig. 5. Block diagram of the proposed TMCIC algorithm.

denote the texture feature of $B(i)$ where $F_{B(i)}(j)$ is defined as follows:

$$F_{B(i)}(j) = \frac{\sum_m \left| W^j_{B(i)}(P_m) \right|}{\text{Size}\left(W^j_{B(i)} \right)} \tag{10}$$

where $W^j_{B(i)}(P_m)$ denotes the coefficient at position P_m in $W^j_{B(i)}$ and $|x|$ denotes the absolute value of x; Size$(W^j_{B(i)})$ denotes the number of coefficients in the subband $W^j_{B(i)}$.

By (10), each block has its own texture feature. To classify the most representative texture features from $\{F_{B(i)} | 0 \le i \le N_B - 1\}$, based on the vector quantization technique, we quantize these N_B texture features into $N_C (\ll N_B)$ texture features. Empirically, N_C is set to 48. Here, an efficient and robust method, namely the minimax partial distortion competitive learning (MMPDCL) method [16], is adopted for vector quantization. Collecting these N_C quantized texture features, our proposed training stage constructs a set of N_C representative texture features denoted by the set $F^* = \{F_i(j)^* | 0 \le i \le N_C - 1, 0 \le j \le 12\}$. Further, according to the quantized texture features F^*, the training blocks B can be grouped into N_C clusters $\{G_i | 0 \le i \le N_C - 1\}$ and each cluster G_i contains N_{Ci} blocks with the similar texture feature.

As shown in Fig. 6, after all training blocks have been clustered by using the MMPDCL method, we then construct the multiple-template $T^*_M = \{T^*_{Mi} | 0 \le i \le N_C - 1\}$ for the N_C representative texture features $F^* = \{F^*_i | 0 \le i \le N_C - 1\}$ by using our proposed multiple-template construction procedure. Let N_T be the size of each constructed template T^*_{Mi}, where T^*_{Mi} is a N_T-pixel template, and then a search template T_E (see Fig. 4) consisting of $(2w+1) \times h + w$ offsets $\{O^E_j | 0 \le j \le (2w+1) \times h + w - 1\}$ is used to determine the candidates for selecting N_T offsets for each constructed template T^*_{Mi}. Empirically, N_T is set to 12. Our proposed multiple-template construction procedure is represented by the following procedure TEMP-CONST.

PROCEDURE TEMP-CONST

1 **for** $i \leftarrow 0$ **to** $N_C - 1$ **do**
2 $T^*_{Mi} \leftarrow \{\phi\}$
3 $T_E \leftarrow \{O^E_m | 0 \le m \le (2w+1) \times h + w - 1\}$
4 **for** $j \leftarrow 1$ **to** N_T **do**
5 $E_{\min} \leftarrow \infty$/*Initialize the minimal entropy $E_{\min}$ to ∞*/
6 Create 2^j contexts $\{C_m | 0 \le m \le 2^j - 1\}$
7 **for** each offset $O^E_n \in T_E$ **do**
8 $T' \leftarrow T^*_{Mi} \cup O^E_n$
9 **for** each block $B(l) \in G_i$ **do**
10 Using T', assign each pixel to one of $\{C_m | 0 \le m \le 2^j - 1\}$
11 Calculate the expected entropy E of $\{C_m | 0 \le m \le 2^j - 1\}$/*see (12)*/
12 **if** $E < E_{\min}$ **then**
13 $\{ E_{\min} \leftarrow E$
14 $O^* \leftarrow O^E_n \}$
15 $T^*_{Mi} \leftarrow T^*_{Mi} \cup O^*$
16 $T_E \leftarrow T_E - O^*$

In the TEMP-CONST procedure, we use the variable i ranged from 0 to $N_C - 1$ in line 1 to select a texture feature F^*_i. For the selected texture feature F^*_i, the corresponding cluster of blocks

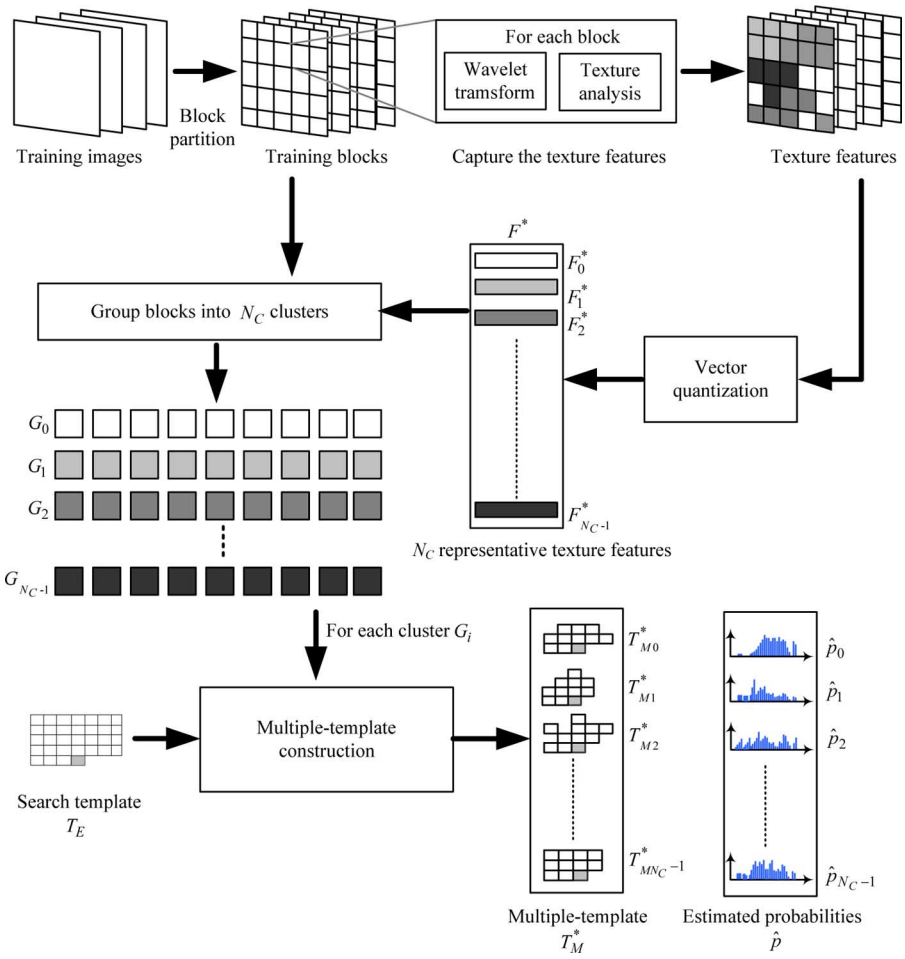

Fig. 6. Block diagram of the proposed training stage.

G_i is used for constructing the template T_{Mi}^*. In line 2, the template T_{Mi}^* corresponding to the texture feature F_i^* is initialized as an empty set. A set of $(2w+1)\times h+w-1$ offsets are assign to the search template T_E (see line 3). The template T_{Mi}^* is constructed progressively by performing the loop containing lines 4-16. In line 4, we use the variable j to increase the template size from 1 to N_T progressively. For each loop iteration, the jth offset in the constructed template T_{Mi}^* is determined from the search template T_E according to the expected entropy obtained by (12). In line 5, the initial minimal entropy value $E_{\min}$ is set to ∞. In line 6, 2^j contexts $\{C_m|0 \le m \le 2^j-1\}$ are created. In line 7, an offset O_n^E is selected from T_E each time, and then a new j-pixel template T' is constructed by uniting the template T_{Mi}^* with the new offset O_n^E (see line 8). For convenience, we use $\{O'_m|0 \le m \le j-1\}$ to represent these j offsets in the new template T'. Let $B_{P_{Bt}}(l)$ denote the pixel at position $P_{Bt} = \{x_{Bt}, y_{Bt}\}$ in the block $B(l), 0 \le x_{Bt} \le W_B$ and $0 \le y_{Bt} \le H_B$. For each $B_{P_{Bt}}(l)$, we utilize T' to compute the address of the context for the pixel $B_{P_{Bt}}(l)$ and the address of $B_{P_{Bt}}(l)$ is given by

$$k = \sum_{m=0}^{j-1} 2^m B_{Q_m}(l) \tag{11}$$

where Q_m denotes the position $P_{Bt} + O'_m$. Since each block is treated separately, when considering the pixels at block boundaries, we set $B_{Q_m}(l) = 0$ if Q_m is not within $B(l)$. After all pixels in the cluster of blocks G_i have been assigned to the corresponding contexts, in line 11, the expected entropy E of the contexts $\{C_m|0 \le m \le 2^j-1\}$ is computed by

$$E = -\sum_{m=0}^{2^j-1}\left[p(C_m)\sum_{b=0}^{1} p(b|C_m)\log_2(p(b|C_m))\right]. \tag{12}$$

We use O^* to denote the offset which causes the current minimal entropy $E_{\min}$ before O_n^E has been selected to compute the entropy E. In line 12, the entropy E is compared to the current minimum entropy $E_{\min}$. If $E < E_{\min}$, the compression performance caused by selecting the offset O_m^E is better than that caused by selecting the offset O^*. Thus, we set $O^* = O_m^E$ and $E_{\min} = E$ in lines 13-14 to record the current best selection of offsets. After all offsets in T_E have been checked, the jth best offset is recorded in O^* and it is inserted into T_{Mi} (see line 15). In line 16, the selected offset O^* is taken from the search template T_E to avoid checking O^* again in next iteration. By performing the loop consisting lines 4-16, the N_T-pixel template T_{Mi}^* can be constructed iteratively. Returning to line 1, we increase the variable i by 1 to construct the template T_{Mi+1}^* for the texture feature F_{i+1}^*. Continuing the loop from line 1 to line 16, finally the multiple-template T_M^* can be constructed.

Since the general arithmetic coding starts with a uniform probability distribution, the compression performance will be

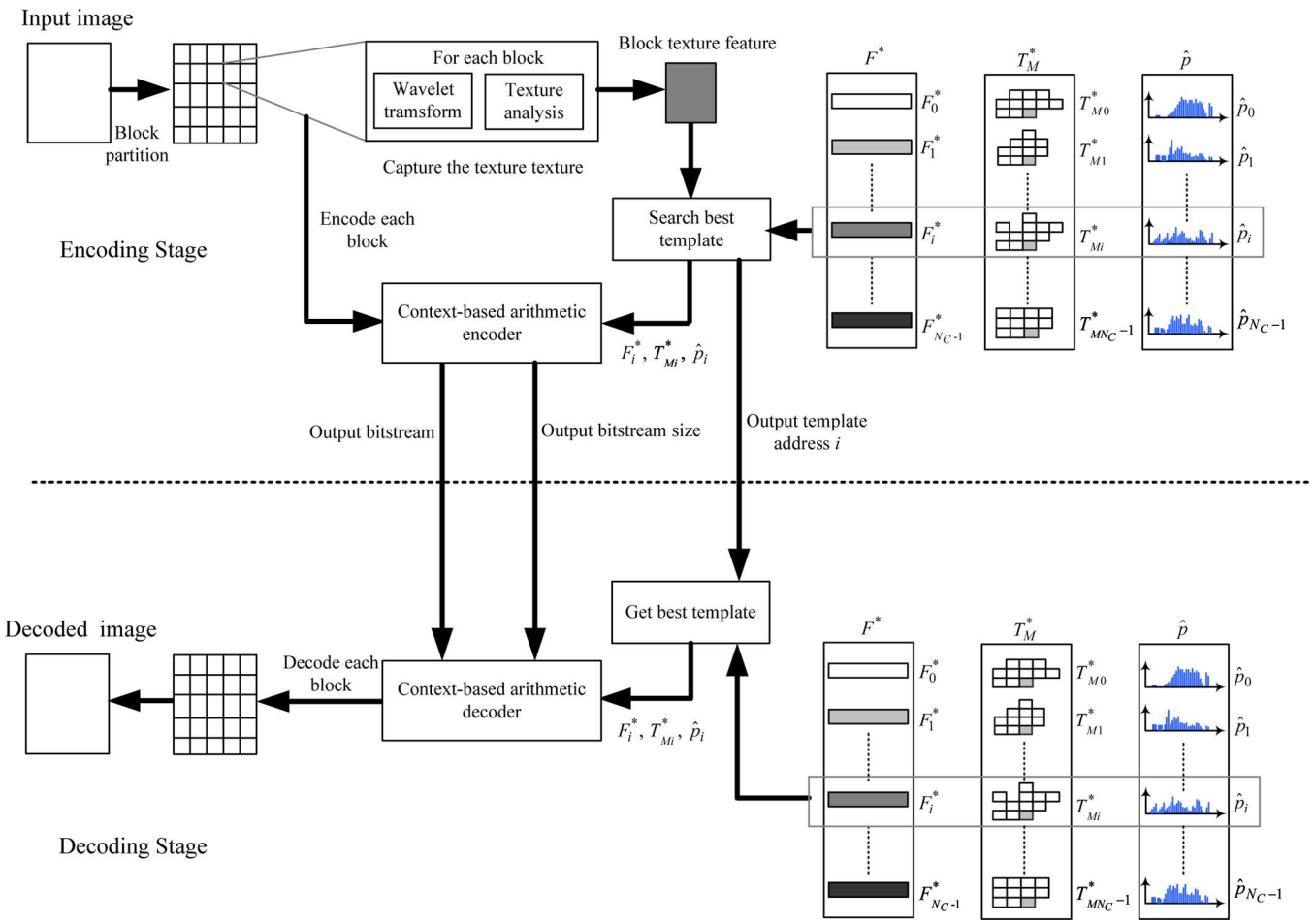

Fig. 7. Block diagram of the proposed encoding and decoding stage.

degraded in the early encoding stage. In order to resolve this problem, the probability of zeros (or ones) in each context determined by T^*_{Mi} is estimated using the nonuniform probability distribution to initialize the arithmetic coding. In line 11, the probability $p(0|C_m)$ can be obtained by computing the entropy via (12). Thus, for the constructed template T^*_{Mi}, $\hat{p}_i(0|C_m)$ is used to estimate the probability of zeros in the context C_m and in lines 12-14, the original three statements are modified into the following six lines to estimate the probability $\hat{p}_i(0|C_m)$.

---

12 **if** $E < E_{\min}$ **then**

13 $\{E_{\min} \leftarrow E$

14(a) $O^* \leftarrow O_n$

14(b) **if** $j = N_T$

14(c) **for** $m \leftarrow 0$ to $2^{N_T} - 1$

14(d) $\hat{p}_i(0|C_m) \leftarrow p(0|C_m)\}$

After the estimated probability $\hat{p}_i(0|C_m)$ is obtained, $\hat{p}_i(1|C_m)$ can be calculated by $1 - \hat{p}_i(0|C_m)$. For convenience, let $\hat{p}_i = \{\hat{p}_i(0|C_m)|0 \leq m \leq 2^{N_T} - 1\}$ denote the set of estimated probabilities for the 2^{N_T} contexts determined by the template T^*_{Mi}. In other words, for a block whose texture feature belongs to F^*_i, the template T^*_{Mi} is selected to code this block and the probabilities of zeros in the contexts $\{C_m|0 \leq m \leq 2^{N_T} - 1\}$ can be estimated by $\hat{p}_i$. Further, for the constructed multiple-template $T^*_M = \{T^*_{Mi}|0 \leq i \leq N_C - 1\}, \hat{p} = \{\hat{p}_i|0 \leq i \leq N_c - 1\}$ is used to denote the set of all estimated probabilities of zeros in the contexts determined by T^*_M.

Although our proposed training stage takes much time to obtain the most representative texture features F^*, the multiple-template T^*_M, and the estimated probabilities $\hat{p}$, it only needs to perform the training stage once and the training results can be reused again and again to code error diffused-images repeatedly. For example, given a block in any input error-diffused image, we first obtain its texture feature and then find the best matched texture feature F^*_i from the set F^*. According to the matched F^*_i, we select T^*_{Mi} and $\hat{p}_i$ as the best template and the corresponding estimated probabilities, respectively, for encoding this block. Therefore, the time required in our proposed training stage can be ignored.

B. Encoding and Decoding Stages

As shown in Fig. 7, the encoding stage divides the input error-diffused image into $W_B \times H_B$ blocks and each block is encoded independently. Since the encoding of each block is independent, for convenience, we define the block $B(i)$ to be the ith encoded block in the input image. Running the four-level CDF 9/7 wavelet transform on $B(i)$, the 13 subbands

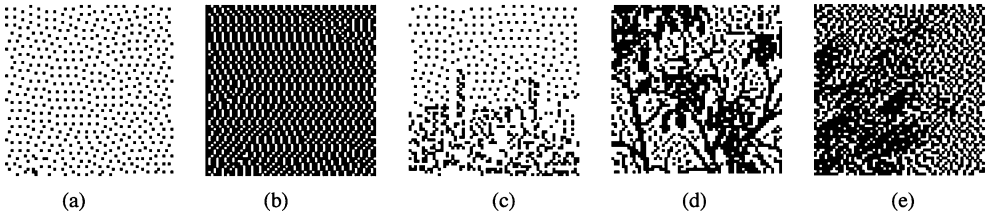

Fig. 8. Five 64×64 blocks with different texture features.

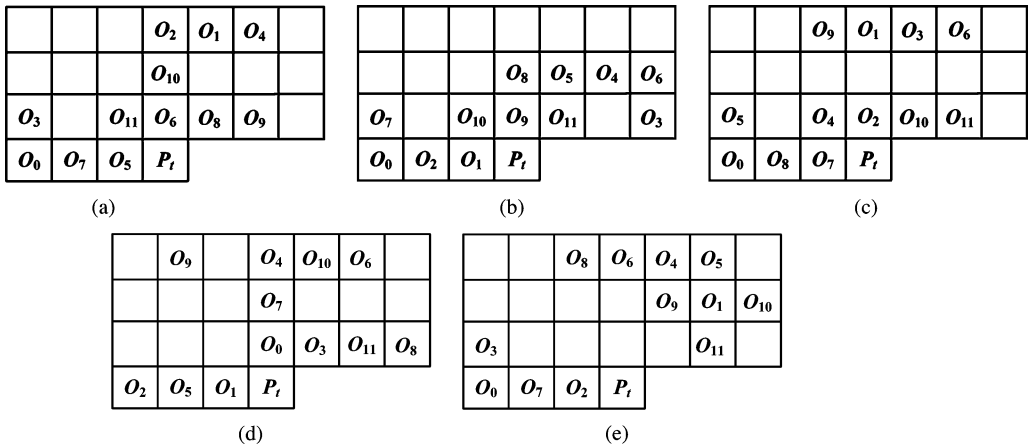

Fig. 9. Five best templates selected by our proposed TMCIC algorithm for Fig 8.

$\{W^i_{B(i)} | 0 \le i \le 12\}$ of $B(i)$ are used to calculate the texture feature $F_{B(i)} = \{F_{B(i)}(j) | 0 \le j \le 12\}$ by (10). Comparing $F_{B(i)}$ to each texture feature of F^*, the address of the best template T^*_{Ck} for $B(i)$ can be determined by

$$k = \arg\min_k \sum_{j=0}^{12} \left| F_{B(i)}(j) - F^*_k(j) \right|^2. \tag{13}$$

To improve the compression performance of context-based arithmetic coding, the set of estimated probabilities $\hat{p_k}(1 - \hat{p_k})$ is used to initialize the number of zeros (ones) in the contexts $\{C_l | 0 \le l \le 2^{N_T}\}$. Let $N_{C_l}(0)$ and $N_{C_l}(1)$ denote the initial number of ones and zeros in the context C_l. According to $\hat{p_k}(0|C_l)$, $N_{C_l}(0)$ and $N_{C_l}(1)$ can be calculated by

$$\begin{aligned} N_{C_l}(0) &= \lfloor \hat{p_k}(0|C_l) \times \beta \rfloor + 1 \\ N_{C_l}(1) &= \lfloor (1 - \hat{p_k}(1|C_l)) \times \beta \rfloor + 1 \end{aligned} \tag{14}$$

where β is set to 50 empirically. From T^*_{Ck}, $\{N_{C_l}(0) | 0 \le l \le 2^{N_T}\}$, and $\{N_{C_l}(1) | 0 \le l \le 2^{N_T}\}$, we encode each pixel in the block $B(i)$ in raster scan order. Let $B_{P_{Bt}}(i)$ denote the current pixel at position P_{Bt} in the block $B(i)$. By the same way as in (11), we assume that $B_{P_{Bt}}(i)$ is assigned to the contexts C_m, and then the probability of $B_{P_{Bt}}(i)$ in C_m can be calculated by (15), shown at the bottom of the page, where $N^t_{C_m}(0)$ $(N^t_{C_m}(1))$ denotes the number of zeros (ones) encoded in context C_m before encoding $B_{P_{Bt}}(i)$. After all pixels in $B(i)$ have been encoded, our proposed encoding stage records the bitstream D_i obtained by the arithmetic coding, the length of the bistream L_{D_i}, and the template address k, and then the next block is selected to encode. The above block-based encoding process is performed successively until all blocks in the input image are encoded.

In the decoding stage, we first read the bitstream size L_{D_i}, and then the bitstream D_i with size L_{D_i} is read for decoding the block $B(i)$. The best template T^*_{Ck} used in $B(i)$ is also selected by reading the recorded template address k. The number of ones and zeros in each context C_l can be initialized by (14). Thus, all pixels in $B(i)$ can be decoded from the bitstream D_i by using the context-based arithmetic decoding, and then the data related to the next block will be read for decoding. Performing the above block-based decoding process successively, all blocks in the image can be decoded by our proposed decoding stage.

IV. Experimental Results

In this section, the results of our proposed training stage are given in the first subsection to demonstrate that the constructed multiple-template can reflect the most representative texture features. Next, the compression performance comparison among the JBIG, the BACIC algorithm, the PACIC algorithm,

$$p(B_{P_{Bt}}(i))|C_m) = \begin{cases} p(0|C_m) = \frac{N^t_{C_m}(0) + N_{C_m}(0)}{N^t_{C_m}(0) + N_{C_m}(0) + N^t_{C_m}(1) + N_{C_m}(1)}, & \text{if } B_{P_{Bt}}(i) = 0 \\ p(1|C_m) = 1 - p(0|C_m), & \text{otherwise} \end{cases} \tag{15}$$

TABLE I
SIZES OF THE BITSTREAMS IN TERMS OF BYTES OBTAINED BY CODING THE BLOCKS WITH DIFFERENT TEMPLATES

	$B(0)$	$B(1)$	$B(2)$	$B(3)$	$B(4)$
$T^*_{M_0}$	140	190	356	595	585
$T^*_{M_1}$	330	99	495	661	576
$T^*_{M_2}$	189	183	320	484	486
$T^*_{M_3}$	278	274	355	483	545
$T^*_{M_4}$	226	153	350	528	433

TABLE II
COMPRESSION RATIOS OF THE JBIG, THE BACIC ALGORITHM, THE PACIC ALGORITHM, THE FACIC ALGORITHM, AND OUR PROPOSED TMCIC ALGORITHM

Image	JBIG	BACIC	PACIC	FACIC	TMCIC
Fig. 10(a)	1.19	1.27	1.26	1.44	1.55
Fig.10(b)	1.24	1.30	1.30	1.51	1.59
Fig. 10(c)	1.04	1.06	1.06	1.13	1.17
Fig. 10(d)	1.22	1.28	1.25	1.42	1.47
Fig. 10(e)	1.20	1.19	1.22	1.41	1.45
Fig. 10(f)	1.72	1.81	1.83	2.13	2.08
Fig. 10(g)	1.40	1.43	1.39	1.69	1.73
Fig. 10(h)	1.13	1.19	1.23	1.45	1.49
Fig. 10(i)	1.27	1.28	1.23	1.39	1.42
Fig. 10(j)	1.09	1.11	1.15	1.25	1.29
Fig. 10(k)	1.16	1.18	1.17	1.38	1.42
Fig. 10(l)	1.23	1.29	1.25	1.41	1.36
Fig. 10(m)	1.32	1.41	1.44	1.64	1.66
Fig. 10(n)	1.44	1.55	1.60	2.11	1.88
Fig. 10(o)	1.54	1.53	1.55	1.80	1.80
Fig. 10(p)	1.54	1.50	1.55	1.99	1.97
Fig. 10(q)	1.41	1.53	1.61	2.13	1.89
Fig. 10(r)	1.18	1.21	1.22	1.43	1.40
Fig. 10(s)	1.32	1.43	1.45	1.92	1.76
Fig. 10(t)	1.22	1.33	1.38	1.79	1.67
Average	1.29	1.34	1.36	1.62	1.60

the FACIC algorithm, and our proposed TMCIC algorithm are demonstrated in the second subsection. All the experiments are implemented by using the personal computer with Intel Celeron 2.8-GHz CPU and the Borland C++ builder 6.0 programming language.

A. Results of Our Proposed Training Stage

In the implementation of our proposed training stage, the Jarvis error diffusion scheme [17] is performed on sixty 768×512 real images [18] to obtain the input error-diffused training images. Dividing each error-diffused training image into 96 training blocks, each with size 64×64, we can totally obtain 5760 training blocks. Empirically, the number of the most representative texture features N_C is set to 48. Thus, we can obtain 48 texture features and the training blocks are grouped into 48 clusters for constructing the multiple-template.

TABLE III
ENCODING TIME OF THE JBIG, THE BACIC ALGORITHM, THE PACIC ALGORITHM, THE FACIC ALGORITHM, AND OUR PROPOSED TMCIC ALGORITHM IN TERMS OF SECONDS

Image	JBIG	BACIC	PACIC	FACIC	TMCIC
Fig. 10(a)	0.024	0.110	0.044	7.062	0.186
Fig. 10(b)	0.025	0.107	0.045	7.705	0.187
Fig. 10(c)	0.023	0.111	0.047	8.016	0.193
Fig. 10(d)	0.023	0.107	0.044	8.735	0.188
Fig. 10(e)	0.022	0.107	0.044	8.188	0.189
Fig. 10(f)	0.019	0.094	0.034	9.188	0.179
Fig. 10(g)	0.020	0.105	0.043	7.828	0.187
Fig. 10(h)	0.021	0.110	0.045	8.078	0.187
Fig. 10(i)	0.020	0.106	0.045	7.984	0.188
Fig. 10(j)	0.021	0.107	0.043	7.705	0.192
Fig. 10(k)	0.021	0.110	0.046	8.172	0.201
Fig. 10(l)	0.063	0.408	0.151	63.343	0.662
Fig. 10(m)	0.062	0.401	0.140	59.734	0.643
Fig. 10(n)	0.204	1.490	0.494	284.750	2.345
Fig. 10(o)	0.200	1.500	0.484	275.359	2.344
Fig. 10(p)	0.230	1.505	0.490	257.453	2.320
Fig. 10(q)	0.211	1.507	0.495	281.203	2.337
Fig. 10(r)	0.231	1.521	0.586	297.578	2.430
Fig. 10(s)	0.220	1.544	0.534	296.532	2.364
Fig. 10(t)	0.227	1.561	0.550	278.021	2.376
Average	0.094	0.630	0.220	109.131	0.995

In our experiment, the proposed training stage spends 28 minutes on the texture classification and the multiple-template construction. Once the texture classification and the multiple-template construction have been done, their results can be reused again and again. Therefore, the time spent on the training stage is not considered in the encoding time. Fig. 8 shows five 64×64 blocks with different texture features. From our training results, the five best templates as shown in Fig. 9(a)–(e) are selected to encode Fig. 8(a)–(e), respectively.

For each template in Fig. 9, O_i denotes the ith offset selected by the TEMP-CONST procedure which has been described in Section III-A. $\{B(i)|0 \le i \le 4\}$ denote the blocks as shown in Fig. 8(a)–(e) and the best selected templates as shown in Fig. 9(a)–(e) are denoted by $\{T^*_{M_j}|0 \le j \le 4\}$, respectively. In order to demonstrate that the selected template T^*_{Mi} can efficiently reflect the texture features of $B(i)$, some experiments are given as follows. For each block $B(i)$, we encode $B(i)$ five times by using the templates $T^*_{M_0}, T^*_{M_1}, \ldots$, and $T^*_{M_4}$, respectively. Let D_{ij} denote the bitstream obtained by encoding the block $B(i)$ using the template $T^*_{M_j}$, and $L_{D_{ij}}$ denotes the size of D_{ij}. Thus, totally, we can obtain 25 bitstreams $\{D_{ij}|0 \le i, j \le 4\}$ and the sizes of these 25 bitstreams in terms of bytes are shown in Table I. Each entry in Table I denotes the bitstream size $L_{D_{ij}}$. From the bitstream sizes shown in Table I, it is observed that for block $B(i)$, the condition $L_{D_{ii}} < L_{D_{ij}}$ is held when $i \neq j$. This observation reveals that the best selected template T^*_{Mi} efficiently reflects the texture features of the block $B(i)$. In other words, the multiple-template constructed by our

Fig. 10. Twenty testing images for evaluating the performance of different compression algorithms.

proposed training stage provides a template set in which the best template can be selected based on the texture feature of each input block.

B. Compression Performance Comparison

In this subsection, some experiments are carried out to compare the performance among the JBIG, the BACIC algorithm, the PACIC algorithm, the FACIC algorithm, and our proposed TMCIC algorithm. For easy implementation, we use the basic arithmetic coding algorithm [19] to implement our proposed TMCIC algorithm. For speeding up the encoding-time of our proposed algorithm, maybe the QM-coder used in JBIG and the MQ-coder used in JBIG2 [20] can be adopted to achieve this goal.

In our experiments, 20 Jarvis error-diffused images as shown in Fig. 10 are used to evaluate the compression performance and each of the 20 testing images is not contained in the sixty training images used in the training stage. Among these 20 testing images, thirteen 512×512 images are shown in Fig. 10(a)–(k); two 1024×1024 images are shown in Fig. 10(l) and Fig. 10(m); seven 2560×1600 images [21] are shown in Fig. 10(n)–(t).

After running the above five concerned algorithms on the error-diffused images as shown in Fig. 10, the compression ratio comparison and the encoding time comparison are shown in Tables II and III, respectively. From Tables II and III, it is observed that with a little encoding time degradation, 0.365, 0.901, and 0.775 s on average, the average compression improvement ratios of our proposed TMCIC algorithm over the JBIG, the BACIC algorithm, and the PACIC algorithm are $(1.60 - 1.29/1.29)(= 24\%), (1.60 - 1.34/1.34)(= 19.4\%)$, and $(1.62 - 1.36/1.36)(= 17.6\%)$, respectively. Further, the encoding time required in the previous FACIC algorithm is 109.131 s on average while our proposed algorithm takes 0.995 s; the average compression ratio of our proposed TMCIC algorithm, 1.60, is quite competitive to that of the FACIC algorithm, 1.62.

Besides the compression ratio, the issue on the near-lossless, progressive and scalable coding is also important for some applications. The JBIG provides the multilayer coding scheme, from the lower image resolution layer to higher image resolution layer, to support the above three functionalities. The BACIC algorithm cannot provide the the near-lossless, progressive and scalable coding since it utilizes one-pass coding scheme with the conventional context template. The PACIC algorithm [12] and its extended version [13] have the above three functionalities due to adopting the concept of S-value and C-value. The input

image is scaled into a quarter of the original size by using the S-value and this scaling pass can be performed several times if necessary. In addition, since the S-value can be used to help the prediction of the C-value, the PACIC algorithm and its extended version have better compression performance when compared to the JBIG and the BACIC algorithm. Due to lacking the multilayer concept, such as the S-value and C-value, our proposed algorithm has no such three functionalities. The main advantage of our proposed TMCIC algorithm is that its compression ratio is quite competitive to that of the FACIC algorithm and the time required in our proposed TMCIC algorithm is much less than the that in the FACIC algorithm. In summary, the PACIC algorithm and its extended version can work well in the near-lossless, progressive and scalable coding applications. However, our proposed TMCIC algorithm can achieve better performance in the low bitrate environment.

V. Conclusion

In this paper, our proposed TMCIC algorithm has been presented for lossless compression of error-diffused images. Based on the training blocks, our proposed TMCIC algorithm can construct the multiple-template in which the best template can be selected to reflect the texture feature of each input block. This leads to high compression performance. Experimental results demonstrate that with a little encoding-time degradation, our proposed TMCIC algorithm has 24%, 19.4%, and 17.6% average compression improvement ratios when compared to the JBIG, the BACIC algorithm, and the PACIC algorithm, respectively. Further, the encoding time required in the previous FACIC algorithm is 109.131 s on average while our proposed algorithm takes 0.995 s; the average compression ratio of our proposed TMCIC algorithm, 1.60, is quite competitive to that of the FACIC algorithm, 1.62.

Because the binary patterns in text images are dependent on the font types and font sizes, this causes the categories of texture features in the text images are much more than those in the error-diffused images. Thus, our proposed training stage is so suitable for text images as it is for error-diffused images. We can modify our proposed texture classification method to segment the document image into the error-diffused image part and the text part. Consequently, our proposed TMCIC algorithm can be used to compress the error-diffused image part; some existing methods, such as the pattern-matching techniques used in JBIG2 [20], can be used to compress the text part.

References

[1] *Standardization of Group 3 Facsimile Apparatus for Document Transmission*, ITU-T Recommendation T.4, 1980.

[2] *Facsimile Coding Schemes and Coding Control Function for Group 4 Facsimile Apparatus*, ITU-T Recommendation T.6, 1984.

[3] G. R. Robertson, M. F. Aburdene, and R. J. Kozick, "Differential block coing of bilevel images," *IEEE Trans. Image Process.*, vol. 5, no. 9, pp. 1368–1370, Sep. 1996.

[4] M. D. Swanson and A. H. Tewfik, "A binary wavelet decomposition of binary images," *IEEE Trans. Image Process.*, vol. 5, no. 12, pp. 1637–1650, Dec. 1996.

[5] M. N. Gurcan, O. N. Gerek, and A. E. Cetin, "Binary morphological subband decomposition for image coding," in *Proc. IEEE Signal Process. Int. Symp. Time-Frequency and Time-Scale Analysis*, 1996, pp. 357–360.

[6] J. Rissanen and G. G. Langdon, Jr., "Universal modeling and coding," *IEEE Trans. Inf. Theory*, vol. 27, no. 1, pp. 12–23, Jan. 1981.

[7] J. Rissanen, "A universal data compression system," *IEEE Trans. Inf. Theory*, vol. IT-29, no. 5, pp. 656–664, Sep. 1983.

[8] ——, "Complexity of strings in the class of Markov sources," *IEEE Trans. Inf. Theory*, vol. 32, no. 5, pp. 526–532, Jul. 1986.

[9] *Coded Representation of Picture and Audio Information-Progressive Bi-Level Image Compression*, ISO/IEC Int. Std. 11544, 1993.

[10] M. D. Reavy and C. G. Boncelet, "An algorithm for compression of bilevel images," *IEEE Trans. Image Process.*, vol. 10, no. 5, pp. 669–676, May 2001.

[11] K. Nguyen-Phi and H. Weinrichter, "A new binary source coder and its application in bi-level image compression," in *Proc. GLOBALCOMM*, 1996, vol. 3, pp. 1483–1487.

[12] C. S. Lee and H. Park, "Near-lossless/lossless compression of error-diffused images using a two-pass approach," *IEEE Trans. Image Process.*, vol. 12, no. 2, pp. 170–175, Feb. 2003.

[13] C. S. Lee and H. Park, "Progressive coding of error-diffused bilevel images," *J. Electron. Imag.*, vol. 12, pp. 173–178, Jan. 2003.

[14] B. Martins and S. Forchhammer, "Tree coding of bilevel image," *IEEE Trans. Image Process.*, vol. 7, no. 4, pp. 517–528, Apr. 1998.

[15] A. Cohen, I. Daubechies, and J. C. Feauveau, "Biorthogonal bases of compactly supported wavelets," *Commun. Pure Appl. Math.*, vol. 45, pp. 485–560, June 1992.

[16] C. Zhu and L. M. Po, "Minimax partial distortion competitive learning for optimal codebook design," *IEEE Trans. Image Process.*, vol. 5, no. 10, pp. 1400–1409, Oct. 1998.

[17] R. Ulichney, *Digital Halftoning*. Cambridge, MA: MIT Press, 1987.

[18] [Online]. Available: http://www.systems.caltech.edu/mese/halftone/.

[19] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, pp. 520–540, Jun. 1987.

[20] *JBIG2 Final Draft International Standard*, ISO/IEC JTC1/SC29/WG1 N1545, 1999.

[21] [Online]. Available: http://interfacelift.com/wallpaper/.

Yong-Huai Huang received the B.S. degree in information management from Aletheia Universit, Taipei, Taiwan, R.O.C., and the M.S. degree in computer science and information engineering from National Taiwan University of Science and Technology University, Taipei, where he is currently pursuing the Ph.D. degree.

His research interests include image processing, image compression, and algorithms.

Kuo-Liang Chung (SM'01) received the B.S., M.S., and Ph.D. degrees in computer science and information engineering from the National Taiwan University, Taipei, Taiwan, R.O.C., in 1982, 1984, and 1990, respectively.

In 1986, he completed his military service. From 1986 to 1987, he was a Research Assistant with the Institute of Information Science, Academic Sinica, Taiwan. He was a Chairman with the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, from 2003 to 2006. His research interests include image/video compression, image/video processing, pattern recognition, coding theory, algorithms, and multimedia applications.

Prof. Chung received the Distinguished Research Award (2004–2007) from the National Science Council, Taiwan.