A Lossless Coding Scheme Using Adaptive Predictors and Arithmetic Code Optimized for Each Image

Ichiro Matsuda, Yuji Umezu, Nao Ozaki, Joji Maeda, and Susumu Itoh
Faculty of Science and Technology, Tokyo University of Science, Noda, 278-8510 Japan

SUMMARY

A highly efficient lossless encoding method for static images is proposed. In this method, multiple linear predictors are created for each image and adaptive prediction that responds to the local structure of images such as edges and textures is achieved by switching between these predictors at the block level. Furthermore, the probability density functions of the prediction errors are categorized by context modeling and modeled by generalized Gaussian functions, and adaptive arithmetic encoding of the prediction errors is performed by using probability tables that are generated for each pixel from this model. Parameters that are needed in the coding such as the prediction coefficients, the predictor selection data for each block, and the shapes of the generalized Gaussian functions are optimized by repeatedly minimizing a cost function that includes the code length of the parameters themselves in addition to the code length of the prediction errors that are calculated from the probability model above, and the parameters are then encoded separately as side data for each image. A procedure is introduced to improve prediction accuracies by using quadtree segmentation to segment the image into variable-sized blocks between which the predictor can change. Coding experiments are conducted and the proposed method is found to produce coding rates of 6 to 44% lower than the international standard JPEG-LS method, with the proposed method achieving superior coding performance that surpasses existing coding methods for all of the images used in the experiments. © 2007 Wiley Periodicals, Inc. Syst Comp Jpn, 38(4): 1–11, 2007; Published online in Wiley InterScience (www.interscience.wiley.com). DOI 10.1002/scj.20630

Key words: lossless encoding; linear predictor; context modeling; arithmetic coding; variable block size adaptive predictor.

1. Introduction

An algorithm called predictive coding is widely used in the field of lossless coding to compress image data without introducing any distortions. Typical predictive encoders consist of a predictor, which predicts the value of the current pixel from the values of several of the neighboring pixels that have already been encoded (reference pixels), and an entropy encoder, which assigns codes to error components that have had their spatial redundancy removed by the predictor [1]. In particular, several adaptive prediction methods have been proposed that vary the weightings of reference pixels (prediction coefficients) in response to edges, textures, and other localized structures in the image, as these are important elements that dominate the coding performance (compression rate). These methods can be broadly categorized into schemes that infer the optimal prediction coefficients from neighboring pixels that have already been encoded [2–4], and schemes that determine the prediction coefficients noncausally using the signal from the image that is actually being encoded [5, 6]. Although the former schemes do not need any special side information because only data that has already been encoded is used for predictions, both the encoder and decoder must be equipped with a function to infer the prediction...
coefficients, and these methods therefore suffer from the weakness that the level of complexity of processing in the decoder is the same as in the encoder. Conversely, although the latter schemes generally perform more stably because inference errors do not occur in the prediction coefficients, all of the information regarding the adaptive prediction needs to be available to the decoder, and the amount of side information therefore increases as the precision of the predictor increases. In other words, in these schemes there is a trade-off between the code length of the side information and the degree of increased prediction accuracy obtained from this information, and in order to obtain high coding rates it is therefore important to construct an encoding process by considering the balance between these two factors. Recently, encoding schemes that use sets of multiple prediction coefficients that are optimized to each image [7] have demonstrated superior performance, and attention has therefore focused on the latter class of encoding schemes. Furthermore, in these methods the prediction coefficients only need to be determined by the encoding system, and so in principle the decoding process can execute at high speed. This is a useful property in applications such as image databases where most of the image data is compressed once, but played back and viewed many times.

In consideration of these points, the authors have previously proposed a method in which the linear predictors that are used in the lossless coding are reconfigured for each image [8]. This method presumes block adaptive prediction [5] that can switch between multiple different predictors at the block level, and the coding is optimized by repeatedly alternating between optimizing the predictor configurations and selecting the optimal predictors for each block. During the configuration of the predictors, rather than minimizing the sum of the least squares of the prediction errors, the prediction coefficients that are suitable for the lossless coding are determined by minimizing a cost function that represents the amount of prediction error data. This cost function was derived based on the assumption that the prediction errors follow a Gaussian distribution after context modeling [9] has been applied. Although this means that for most images the coding performance of the described method is better than in the case of using a general minimum mean-square error (MMSE) predictor, optimal results will not necessarily be obtained for images where the assumption of a Gaussian distribution breaks down. Furthermore, the cost function does not include the code length of predictor selection data or the prediction coefficients themselves, and so the method is not optimal when the balance between the code length of the side information and the code length of the prediction errors is also considered.

A method that resolves these problems and obtains relatively superior coding efficiency is therefore investigated. More specifically, this method reduces the mismatch between the cost function and the actual code length by introducing generalized Gaussian functions, which are applicable to a wide range of images, to model the probability density functions of the prediction errors, and by changing the entropy encoder from a variable length code based encoder to an arithmetic code based encoder. The total code length of the image is also designed to be minimized by adding the code length needed for side information such as prediction coefficients to the cost function in addition to the code length of the prediction errors. The precision of the block adaptive prediction is also improved by performing predictor switching on variable-sized blocks based on quad-tree partitioning, instead of the fixed sized blocks that are conventionally used. Finally, a program that implements this method is used to perform encoding experiments to quantitatively evaluate the effectiveness of the method.

2. Encoding Order

In the proposed lossless encoding scheme, encoding is performed by scanning a two-dimensional image signal in raster-scan order. Each small block (for example, 8 x 8 pixels) in the image is categorized into one of M classes such that the linear predictor that is allocated to each pixel depends on the class to which the pixel belongs. For a given pixel $p_0$, the value predicted by the $m$-th predictor ($m = 1, 2, \ldots, M$) is given by the equation

$$\hat{s}(p_0) = \sum_{k=1}^{K} a_m(k) \cdot s(p_k)$$

(1)

where $K$ is the number of reference pixels (prediction order) used in the prediction, $a_m(k)$ ($k = 1, 2, \ldots, K$) is the weighting (prediction coefficient) of reference pixel $p_k$, and $s(p_k)$ is the brightness value of pixel $p_k$. The indices $k$ of the reference pixels are assigned in order of increasing Manhattan metric distance from the current pixel over the region where encoding has already finished. Figure 1 shows the arrangement of the reference pixels when the prediction order is set to $K = 30$.

The prediction error $e = s(p_0) - \hat{s}(p_0)$ that is calculated as the result of the aforementioned block adaptive predictor is adaptively entropy encoded using a method called context modeling. Context modeling is a method for inferring the probability density function $P(e)$ of the prediction error of a pixel from the state (context) of the set of surrounding pixels that have already been encoded. The context modeling in the proposed method uses a characteristic quantity $U$ that is defined from neighboring prediction errors in the same way as in Ref. 9:

$$U = \sum_{k=1}^{12} \frac{1}{\delta_k} \cdot |s(p_k) - \hat{s}(p_k)|$$

(2)
where \( \delta \) is the Euclidean distance between the current pixel \( p_0 \) and the reference pixel \( p_k \). The characteristic quantity \( U \) is defined as the sum of the absolute values of the prediction errors of the surrounding 12 pixels weighted by the reciprocal value of \( \delta \). The characteristic quantity \( U \) is then taken to be an indicator of the ease with which the surrounding pixels can be predicted, and the probability density function \( P(e) \) of the prediction error \( e \) can be thought of as largely depending on the value of \( U \).

The characteristic quantity \( U \) is then quantized into 16 levels using the threshold values \( \{ T_{\text{th}}(1), T_{\text{th}}(2), \ldots, T_{\text{th}}(15) \} \) and the conditional probability density functions \( P(e|n) \) are modeled for each of the contexts \( (n = 1, 2, \ldots, 16) \) that correspond to each of these levels. Because the context of each pixel can be determined uniquely solely from the prediction errors that have already been encoded and from the threshold values, it is possible for the adaptive entropy coding to switch between conditional probability models for the prediction errors \( e \) at the pixel level depending on the context.

### 3. Probability Density Function Model of Prediction Errors and Data Lengths

In Ref. 8, the conditional probability density functions \( P(e|n) \) of the prediction errors \( e \) that were observed separately in each context \( (n = 1, 2, \ldots, 16) \) were assumed to follow a Gaussian distribution with a variance \( \sigma^2 \). Based on this assumption, the data length of the prediction errors can be approximated by the sum of the squares of \( e \) weighted by the reciprocal of the variance \( \sigma^2 \) (i.e., the sum of \( e^2/\sigma^2 \)), and the set of prediction coefficients that minimize this value can easily be determined [8] by solving a normal equation in the same way as for an MMSE predictor [5]. In real images, however, the above assumption may not necessarily hold true, and a problem arises in that the efficiency of the entropy encoder is reduced due to mismatches in the probability density function. Figure 2 shows the distribution of prediction errors found in different contexts in an image denoted Camera and an image denoted Airplane. Compared to the distributions from the Airplane image, which are relatively close to Gaussian functions, the shapes of the distributions from the Camera image exhibit relatively sharp peaks in the neighborhood of \( e = 0 \) where the variances of the contexts were particularly small. The probability density functions are therefore modeled by new generalized Gaussian functions [10] to resolve these discrepancies.

\[
\hat{P}(e|n) = \frac{c_n \cdot \eta(c_n, \sigma_n)}{2 \Gamma(1/c_n)} \cdot \exp \left\{ \frac{-|e(c_n, \sigma_n)\cdot e|^{c_n}}{\sigma_n} \right\},
\]

\[
\eta(c_n, \sigma_n) = \frac{1}{\sigma_n \sqrt{\Gamma(3/c_n)/\Gamma(1/c_n)}}
\]

where \( \Gamma(\cdot) \) is the gamma function, \( \sigma_n \) is the standard deviation, and \( c_n \) is a parameter (shape parameter) that controls the shape of the distribution [11]. The generalized Gaussian function is a Laplacian when \( c_n = 1 \), and a Gaussian when \( c_n = 2 \), as shown in Fig. 3. In this paper, in order to obtain a probability density function model \( P(e|n) \) that is applicable to real images, the value of \( c_n \) is allowed to vary between
In the proposed scheme, the values of the prediction errors $e$ of each class are arithmetically encoded adaptively based on the above conditional probability $P(e|\hat{s}(p_0), n)$. The numerator in Eq. (5) is the value obtained by integrating the probability density function model $\hat{P}(e|n)$ over a region of width $h_e$ centered at $e$ (the area shown shaded in dark gray in Fig. 4), and the denominator is the sum over the values enumerated in Eq. (4) of integrals that are found in the same way as the numerator (the total area of the gray regions in the same diagram). The probabilities $P(e|\hat{s}(p_0), n)$ that are needed during the arithmetic encoding can therefore be calculated at high speeds by determining all of these values in advance using a sampling interval of $h_e$ and referring to a table during the encoding process. The number of bits of the encoded prediction error for each pixel when this adaptive arithmetic encoding is executed can be estimated by the equation

$$L(e|\hat{s}(p_0), n) = -\log_2 P(e|\hat{s}(p_0), n) = \log_2 \left\{ \sum_{s=0}^{255} P(s-\hat{s}(p_0)|n) \right\} - \log_2 P(e|n) \quad (7)$$

The function $L(e|\hat{s}(p_0), n)$ is treated as the data length of prediction errors, and is included in the cost function used in the optimization process described in the next section. The number of calculations needed to evaluate the value of this cost function is greatly reduced by using table look-ups for each of the terms on the right-hand side of Eq. (7).

The sampling positions of the probability density function model $\hat{P}(e|n)$ that corresponds to the fractional part of the prediction value $\hat{s}(p_0)$ can be finely controlled by setting the quantization step width $h_e$ of the prediction value $\hat{s}(p_0)$ to a small value [13]. However, experimental results showed that only slight improvements in coding efficiency were obtained for $h_e < 1/8$, and because

$$e \in \{ s - \hat{s}(p_0) \mid s = 0, 1, \ldots, 255 \} \quad (4)$$

The conditional occurrence probability of a prediction error $e$ when both the prediction value $\hat{s}(p_0)$ and context $n$ are given simultaneously can be represented using a probability density function model $\hat{P}(e|n)$ as follows:

$$P(e|\hat{s}(p_0), n) = \frac{P(e|n)}{\sum_{s=0}^{255} P(s-\hat{s}(p_0)|n)}, \quad (5)$$

$$P(e|n) = \int_{-h_e/2}^{h_e/2} \hat{P}(e + \epsilon|n) \, d\epsilon \quad (6)$$

In the proposed scheme, the values of the prediction errors $e$ of each pixel are arithmetically encoded adaptively based on the above conditional probability $P(e|\hat{s}(p_0), n)$. The numerator in Eq. (5) is the value obtained by integrating the probability density function model $\hat{P}(e|n)$ over a region of width $h_e$ centered at $e$ (the area shown shaded in dark gray in Fig. 4), and the denominator is the sum over the values enumerated in Eq. (4) of integrals that are found in the same way as the numerator (the total area of the gray regions in the same diagram). The probabilities $P(e|\hat{s}(p_0), n)$ that are needed during the arithmetic encoding can therefore be calculated at high speeds by determining all of these values in advance using a sampling interval of $h_e$ and referring to a table during the encoding process. The number of bits of the encoded prediction error for each pixel when this adaptive arithmetic encoding is executed can be estimated by the equation

$$L(e|\hat{s}(p_0), n) = -\log_2 P(e|\hat{s}(p_0), n) = \log_2 \left\{ \sum_{s=0}^{255} P(s-\hat{s}(p_0)|n) \right\} - \log_2 P(e|n) \quad (7)$$

The function $L(e|\hat{s}(p_0), n)$ is treated as the data length of prediction errors, and is included in the cost function used in the optimization process described in the next section. The number of calculations needed to evaluate the value of this cost function is greatly reduced by using table look-ups for each of the terms on the right-hand side of Eq. (7).
the amount of memory required for the reference table described earlier grows if the value of $h_r$ is made smaller than necessary, a value of $h_r = 1/8$ was used in the present research.

4. Optimization of Encoding Parameters

As has been described earlier, the following parameters are needed during the decoding process as side information in conjunction with the prediction errors:

- Prediction coefficients $a_m(k)$
- Predictor (class) selection data $m$
- Context modeling threshold values $T_m(n)$
- Probability density function model shape parameter $c_n$

These parameters are optimized iteratively to reduce the following cost function for each of the images under consideration [14]:

$$ J = \sum_{p_0} L(e | \tilde{s}(p_0), n) + B_a + B_m + B_t \quad (8) $$

where the first term on the right-hand side is the total code length of the prediction errors as represented by Eq. (7), and the second to fourth terms are the code lengths required to define each of the parameters $a_m(k)$, $m$, and $T_m(n)$ listed above. The number of bits after encoding is thus reduced in the proposed scheme by using a cost function that is not simply equat to the code length of the prediction errors, but that also includes the code length of the supplementary data itself. The following sections describe the optimization procedures and encoding methods for each of the parameters. The initial values of each of the parameters are predetermined using the method in Ref. 8. The steps in Sections 4.1 to 4.4 are a single iteration of processing which is executed repeatedly until the value of the cost function $J$ that evaluates the entire image stops decreasing or the iteration count reaches some predefined iteration limit (which is set to 100 iterations in this paper).

4.1. Prediction coefficient optimization

In contrast to the length of prediction error data that is derived in Ref. 8 by assuming a Gaussian distribution, the set of prediction coefficients that minimize the $J$ in Eq. (8) cannot be determined analytically because the $a_m(k)$ are not differentiable. The value of $J$ is therefore gradually reduced by making very small corrections to the individual prediction coefficients within a range of correction values and then searching for the minimum value of $J$ in this range. More specifically, two of the prediction coefficients to be corrected, $a_m(i)$ and $a_m(j)$, are randomly selected and the value of the cost function $J_m(\Delta a_i, \Delta a_j)$ when the correction terms $\Delta a_i$ and $\Delta a_j$ are added to the prediction coefficients is determined by evaluating over the region to which the corresponding predictor is applied (i.e., all of the blocks for which the $m$-th class was selected). The context $n$ of each of the pixels is assumed to remain unchanged by the correction terms $\Delta a_i$ and $\Delta a_j$ during this process, and the number of computations needed to determine the prediction values $\Delta \tilde{s}(p_0)$ is reduced by only recalculating the changes $\Delta \tilde{s}(p_0)$ from the precorrected values instead of recalculating Eq. (1):

$$ \Delta \tilde{s}(p_0) = \Delta a_i \cdot s(p_i) + \Delta a_j \cdot s(p_j) \quad (9) $$

The combinations of correction terms $\Delta a_i$ and $\Delta a_j$ that are used are shown in Fig. 5 as black dots (total of 33 points) on a grid. The grid spacing $h_a$ corresponds to the quantization step width of the prediction coefficients, as is described later. The point that has the minimum value from among the cost values $J_m(\Delta a_i, \Delta a_j)$ that were determined for each of the 33 points is selected, and the prediction coefficients are updated only by the amounts $\Delta a_i$ and $\Delta a_j$ that correspond to the selected point. The above process is repeated a fixed number of times while the indices $(i, j)$ of the coefficients to adjust are changed randomly, and the entire process is then repeated for the remaining predictors ($m = 1, 2, \ldots, M$). The reason that the sets of two coefficients that are used as correction candidates are selected from the set of grid points that lie within a band, as shown in Fig. 5, is that there is generally a tendency for the sum of the prediction coefficients before and after being updated to be preserved, and so the search range can be efficiently narrowed down by imposing the restriction of $|\Delta a_i + \Delta a_j| \leq h_a$.

![Fig. 5. Candidate points in refinement of prediction coefficients $a_m(i)$ and $a_m(j)$.](image-url)
The final prediction coefficients that are determined are linearly quantized in the range of \(-2 \leq c_n(k) \leq 2\), and are arithmetically encoded using a probability table based on the Laplace distribution. The quantization step width \(h_n\) that is used in this procedure is a parameter that affects the calculation precision of predictions as well as the code length \(B_n\) of the prediction coefficients. This is set to \(h_n = 1/64\) in this paper based on the results of an investigation into the relationship between the precision of prediction coefficients and encoding efficiency [15]. By setting the quantization precision of the prediction coefficients to a power of two, the numerical calculations used in the predictions can be constructed using fixed point arithmetic, which not only has the advantage of improving calculation speeds, but also allows the calculation errors that are inherent to computers to be eliminated.

4.2. Class selection optimization

Each of the \(M\) different predictors is tried on each block, and the index \(m\) of the predictor that minimizes the cost function within the block is selected as the class of the block. This selection data is not directly encoded as the value of \(m\) for each block, but is encoded as an index into a lookup table that is updated sequentially using a method called move-to-front (MTF) [16]. Figure 6 shows an example of how the lookup table is updated when the class selection data of the blocks that are enclosed in thick borders \((m = 5, 2, 5, 5, 7, \ldots )\) are encoded using the MTF method. The lookup table holds integer values that correspond to the values of \(m\), and which are reordered so that the values of the upper, left, and upper right blocks appear in that order in the uppermost positions in the table. (If two or more locations have the same value, that value is moved to the top.) By doing this, the indices to the values of \(m\) (the positions within the lookup table) that are to be encoded are generally small values if there is a correlation between the predictors that were selected for neighboring blocks (in the example in Fig. 6, the sequence is 2, 1, 0, 1, 0, \ldots ), and the indices can therefore be encoded efficiently. The probability table that is needed to arithmetically encode the index values is based on the actual frequencies of the indices, and this is downloaded for each image by the decoder. Furthermore, because the cost function in Eq. (8) includes the code length \(B_n\) of the class selection data when this probability table is used, neighboring blocks are more likely to select the same predictor in flat regions and other areas where there are no prominent differences between the properties of each of the predictors, and this further improves the effectiveness of the MTF method [14].

4.3. Context modeling threshold value optimization

The combination of threshold values \(\{Th_m(n)\} (n = 1, 2, \ldots, 15)\) that minimizes the cost function \(J\) is determined over each region of blocks that have the same predictor selection (class). This is achieved by treating the problem as 15 separate threshold value combination optimization problems so that the solutions can be found quickly using dynamic programming methods [8]. Furthermore, the obtained threshold values \(Th_m(n)\) can be treated as a monotonically increasing progression of numbers over \(n\) and so are suitable for arithmetic encoding using a single-sided Laplacian distribution probability model over the differences between the threshold values \(Th_m(n) - Th_m(n - 1)\).

4.4. Shape parameter optimization

For each of the contexts, which are determined from the threshold values found in Section 4.3, the shape parameters \(c_n (n = 1, 2, \ldots, 16)\) of the probability density functions, which are modeled by the generalized Gaussian function given in Eq. (3), are optimized. More specifically, the value of the cost function \(J\) from Eq. (8) is evaluated as the value of \(c_n\) is incremented in steps of 0.2, and the value of \(c_n\) that minimizes the value of the cost function is selected for each context [12]. Because the values of \(c_n\) (16 parameters) are arithmetically encoded using an equal probability, the code length can be taken to be a constant (4 bits per context), and so would not have an effect on the optimization result even if it were included in the cost function in Eq. (8).
5. Variable Block-Size Adaptive Prediction

In the block adaptive prediction used in the proposed scheme, although the code length of the prediction errors can generally be reduced by decreasing the size of blocks between which predictors can be switched, this also increases the code length $B_m$ of the class selection data, as described in Section 4.2, and the overall encoding ratio does not necessarily improve. In Refs. 5, 6, and 8, $8 \times 8$ pixel square blocks were used, as this is a size that is appropriate to all situations from the point of view of the trade-off between the code length of prediction errors and the amount of side information. However, the actual optimal value of block size is thought to be largely dependent on localized properties of the image, and the encoding ratio can therefore be further improved by making the block size variable. This paper thus includes variable-size block adaptive prediction based on quadtree partitioning [17, 18]. More specifically, a maximum block size of $32 \times 32$ pixels is recursively divided into 4 parts down to blocks of $2 \times 2$ pixels, as shown in Fig. 7, and the value of the cost function $J$ is evaluated when the appropriate predictor (class) is allocated to each of these blocks. The block segmentation state is represented by a quadtree in which the $32 \times 32$ pixel block is the root, and nodes (blocks) that are divided to form branches are assigned a flag of "1" and nodes that form terminating leaves are assigned "0." The block sizes are encoded by adaptive binary arithmetic encoding of the flag data, with the layer (which corresponds to the block size) and segmentation state of the surrounding blocks (the number of nodes in the same layer that have the flag "1") used as the context for each node. Furthermore, during encoding of the class selection data, the class numbers of the blocks that contain the pixels A and B that are adjacent to the top left of the block, and the pixel C that is adjacent to the top right corner, as shown in Fig. 8, are used as reference values to utilize the MTF method described in Section 4.2. The minimum value of the cost function $J$ when these code lengths are included is determined, and the combination of block sizes and class selections that correspond to this minimum value are determined for each region of $32 \times 32$ pixels. The resulting block segmentation that minimizes the code length of the entire image can then be determined by repeatedly performing this process instead of the class selection optimization process described in Section 4.2.

6. Evaluation of the Encoding Characteristics

To evaluate the encoding characteristics of the proposed scheme, an encoder/decoder program was written in C. A high-speed multivalue arithmetic coding algorithm known as RangeCoder [19] was used as the entropy encoder. It is generally difficult to determine the appropriate values of the prediction order $K$ and number of predictors $M$ for each image. In order to obtain properties that are superior on average for each image size, these parameters were automatically set based on the following equations, which were derived from preliminary experiments [15]:

$$K = d \cdot (d + 1) \mid d \in \{1, 2, 3, \ldots \} \quad \text{s.t.} \quad d^2 \leq 1.2 \cdot 10^{-4} X + 17.2 < (d+1)^2$$

$$M = \lceil 10.4 \cdot 10^{-5} X + 13.8 \rceil$$

where $X$ is the area (number of pixels) of the image to be encoded. Furthermore, the $d$ in Eq. (10) is a natural number that corresponds to the maximum Manhattan metric distance between the pixel being examined $p_0$ and the reference pixels $p_i$ in Fig. 1. Conversely, the number of contexts used in the adaptive arithmetic encoding of the prediction errors is a parameter for which the optimal value is relatively independent of the image [8], and so is set to the fixed value of 16 in this paper, as was described earlier.

The 13 monochrome (8-bit precision) images shown in Fig. 9 were used to compare the coding rates (bits/pel) of the proposed method to variations of the proposed method where some of the coding parameters had been

![Fig. 7. Example of quadtree-based variable block-size partitioning.](image-url)

![Fig. 8. Reference points of class labels for the Move-To-Front method.](image-url)
changed, and the results are shown in Table 1. The numbers in parentheses in the table show the reduction in coding rate (%) as defined by the equation

$$\left(1 - \frac{\text{encoding rate of proposed method}}{\text{encoding rate of comparative method}}\right) \times 100\%$$  \hspace{1cm} (12)

The method names are made up of three different components, which are delimited by hyphens, that refer to differences in the encoding parameters. The components are defined as follows starting from the first component: whether the block sizes are variable (VS) or are fixed at 8 x 8 pixels (FS); whether arithmetic coding (AR) or Huffman coding (HF) was used as the entropy encoder; and whether the probability density function of the prediction errors was modeled by a generalized Gaussian function (GG) or a Gaussian function (GS). When the probability density function was modeled by a Gaussian function (corresponding to restricting the value of the shape parameter to $c_k = 2.0$), the weighted square error proposed in Ref. 8 was used as the code length of the prediction errors. The method FS-HF-GS is therefore virtually the same as the method in Ref. 8 except that the cost function includes the code length of side information and the values of the parameters $K$ and $M$ differ. The results in Table 1 show that the encoding rates were improved by 0.01 to 0.019 bit/pel by modeling the probability density functions of the prediction errors by generalized Gaussian functions, which have a high degree of freedom. The magnitude of this improvement exhibits a tendency to be much more dependent on the type of image than on differences in the entropy encoder. Figure 10 shows examples of prediction error distributions and the probability density function models of those errors for each of the contexts when the Camera image is encoded by the FS-AR-GG method. The mismatch between the real distribution and the distribution in the model is clearly

Table 1. Comparison of coding rates (bits/pel)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera</td>
<td>$256 \times 256$, $K = 30$, $M = 20$</td>
<td><strong>3.949</strong></td>
<td><strong>3.989</strong> (1.0)</td>
<td><strong>4.043</strong> (2.4)</td>
<td><strong>4.033</strong> (2.1)</td>
<td><strong>4.075</strong> (3.1)</td>
</tr>
<tr>
<td>Couple</td>
<td>$3.888$</td>
<td>3.403 (0.4)</td>
<td>3.442 (1.6)</td>
<td>3.485 (2.8)</td>
<td>3.514 (3.6)</td>
<td></td>
</tr>
<tr>
<td>Noisesquare</td>
<td>$5.270$</td>
<td>5.288 (0.3)</td>
<td>5.359 (1.7)</td>
<td>5.297 (0.5)</td>
<td>5.350 (1.5)</td>
<td></td>
</tr>
<tr>
<td>Airplane</td>
<td>$512 \times 512$, $K = 42$, $M = 41$</td>
<td><strong>5.663</strong></td>
<td>5.667 (0.1)</td>
<td>5.685 (0.4)</td>
<td>5.696 (0.6)</td>
<td>5.707 (0.8)</td>
</tr>
<tr>
<td>Baboon</td>
<td>$4.280$</td>
<td>4.288 (0.2)</td>
<td>4.311 (0.7)</td>
<td>4.322 (1.0)</td>
<td>4.336 (1.3)</td>
<td></td>
</tr>
<tr>
<td>Lenna</td>
<td>$3.889$</td>
<td>3.900 (0.3)</td>
<td>3.922 (0.8)</td>
<td>3.933 (1.1)</td>
<td>3.946 (1.4)</td>
<td></td>
</tr>
<tr>
<td>Lennagrey</td>
<td>$4.199$</td>
<td>4.207 (0.2)</td>
<td>4.226 (0.6)</td>
<td>4.234 (0.8)</td>
<td>4.249 (1.2)</td>
<td></td>
</tr>
<tr>
<td>Peppers</td>
<td>$0.695$</td>
<td>0.755 (0.3)</td>
<td>0.939 (27.6)</td>
<td>1.415 (51.6)</td>
<td>1.601 (57.2)</td>
<td></td>
</tr>
<tr>
<td>Shapes</td>
<td>$2.579$</td>
<td>2.584 (0.2)</td>
<td>2.611 (1.2)</td>
<td>2.627 (1.8)</td>
<td>2.652 (2.8)</td>
<td></td>
</tr>
<tr>
<td>Balloon</td>
<td>$3.815$</td>
<td>3.827 (0.3)</td>
<td>3.869 (1.4)</td>
<td>3.861 (1.2)</td>
<td>3.897 (2.1)</td>
<td></td>
</tr>
<tr>
<td>Barb2</td>
<td>$4.216$</td>
<td>4.224 (0.2)</td>
<td>4.261 (1.1)</td>
<td>4.276 (1.4)</td>
<td>4.300 (2.0)</td>
<td></td>
</tr>
<tr>
<td>Goldhill</td>
<td>$4.207$</td>
<td>4.215 (0.2)</td>
<td>4.241 (0.8)</td>
<td>4.265 (1.4)</td>
<td>4.283 (1.8)</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>$3.828$</td>
<td>3.842 (0.4)</td>
<td>3.887 (1.6)</td>
<td>3.922 (2.6)</td>
<td>3.966 (5.5)</td>
<td></td>
</tr>
</tbody>
</table>

A list of test images.

Fig. 9. Test images.
Fig. 10. Distributions of prediction errors and the corresponding PDF models.

reduced compared to the lower part of Fig. 2 (the model FS-AR-GS). Differences in the statistical properties of the prediction errors between images are therefore thought to have been handled by the probability density function model based on generalized Gaussian functions. Furthermore, the coding rates when arithmetic coding was used compared to those when Huffman coding was used exhibit a large improvement, particularly for the Shapes image. This is because the Shapes image is computer-generated and therefore has extremely low entropy in the prediction errors, and this clearly demonstrates the weakness of variable length codings such as Huffman coding in which each symbol cannot be allocated to less than 1 bit. Furthermore, the proposed method achieved an approximately 0.016 bit/pel lower coding rate compared to the method FS-AR-GG with fixed block sizes, demonstrating the effectiveness of variable block size adaptive prediction based on quadtree segmentation. Figure 11 shows the segmentation of blocks by the proposed method for the Camera image, which
demonstrates the way in which suitable block sizes are selected in response to edges and the degree of complexity of textures in an image.

To objectively evaluate the coding performance of the proposed method, a comparison was next performed against the state of the art lossless coding methods BMF [20], TMW [7], Glicbawls [4], CALIC [21], JPEG-LS [22], and JPEG-2000 [23]. The results are shown in Table 2, with each of the values in the table having the same meanings as those in Table 1. The results for BMF demonstrate the effectiveness of choosing the encoder that offers the best performance for each image from among encoders that have been provided for use with natural images and for use with computer graphics and the results for CALIC demonstrate the effectiveness of using an arithmetic coder as the entropy encoder. JPEG-2000 is a method that was developed with

<table>
<thead>
<tr>
<th>Image</th>
<th>VS-AR-GG (Proposed)</th>
<th>BMF</th>
<th>TMW</th>
<th>Glicbawls</th>
<th>CALIC</th>
<th>JPEG-LS</th>
<th>JPEG 2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera</td>
<td>3.949</td>
<td>4.060 (2.7)</td>
<td>4.098 (3.6)</td>
<td>4.208 (6.2)</td>
<td>4.190 (5.7)</td>
<td>4.214 (8.4)</td>
<td>4.535 (12.9)</td>
</tr>
<tr>
<td>Couple</td>
<td>3.388</td>
<td>3.446 (1.7)</td>
<td>3.446 (1.7)</td>
<td>3.543 (4.4)</td>
<td>3.609 (6.1)</td>
<td>3.699 (8.4)</td>
<td>3.915 (13.5)</td>
</tr>
<tr>
<td>NoiseSquare</td>
<td>5.270</td>
<td>5.522 (0.5)</td>
<td>5.522 (4.9)</td>
<td>5.415 (2.7)</td>
<td>5.443 (3.2)</td>
<td>5.683 (7.3)</td>
<td>5.634 (6.5)</td>
</tr>
<tr>
<td>Airplane</td>
<td>3.591</td>
<td>3.602 (0.3)</td>
<td>3.601 (0.3)</td>
<td>3.668 (2.1)</td>
<td>3.743 (4.1)</td>
<td>3.817 (5.9)</td>
<td>4.013 (10.5)</td>
</tr>
<tr>
<td>Balloon</td>
<td>5.663</td>
<td>5.714 (0.9)</td>
<td>5.714 (1.3)</td>
<td>5.666 (0.9)</td>
<td>5.875 (3.6)</td>
<td>6.037 (6.2)</td>
<td>6.107 (7.3)</td>
</tr>
<tr>
<td>Lena</td>
<td>4.280</td>
<td>4.317 (0.9)</td>
<td>4.300 (0.5)</td>
<td>4.295 (0.3)</td>
<td>4.475 (4.4)</td>
<td>4.607 (7.1)</td>
<td>4.684 (8.6)</td>
</tr>
<tr>
<td>Lena Grey</td>
<td>3.880</td>
<td>3.929 (1.0)</td>
<td>3.908 (0.5)</td>
<td>3.901 (0.3)</td>
<td>4.102 (5.2)</td>
<td>4.238 (8.2)</td>
<td>4.303 (9.6)</td>
</tr>
<tr>
<td>Peppers</td>
<td>4.109</td>
<td>4.214 (1.6)</td>
<td>4.251 (1.2)</td>
<td>4.246 (1.2)</td>
<td>4.421 (5.0)</td>
<td>4.513 (7.0)</td>
<td>4.629 (9.3)</td>
</tr>
<tr>
<td>Shapes</td>
<td>0.685</td>
<td>0.730 (5.1)</td>
<td>0.740 (7.5)</td>
<td>2.291 (70.1)</td>
<td>1.139 (39.8)</td>
<td>1.214 (43.5)</td>
<td>1.926 (64.4)</td>
</tr>
<tr>
<td>Balloon</td>
<td>2.579</td>
<td>2.649 (2.7)</td>
<td>2.649 (2.7)</td>
<td>2.640 (2.3)</td>
<td>2.825 (8.7)</td>
<td>2.964 (11.2)</td>
<td>3.031 (14.9)</td>
</tr>
<tr>
<td>Barb</td>
<td>3.815</td>
<td>3.959 (3.6)</td>
<td>4.084 (6.6)</td>
<td>3.916 (2.6)</td>
<td>4.413 (13.6)</td>
<td>4.691 (18.7)</td>
<td>4.600 (17.1)</td>
</tr>
<tr>
<td>Barb2</td>
<td>4.216</td>
<td>4.276 (1.4)</td>
<td>4.378 (3.7)</td>
<td>4.318 (2.4)</td>
<td>4.560 (6.9)</td>
<td>4.686 (10.0)</td>
<td>4.789 (12.0)</td>
</tr>
<tr>
<td>Goldhill</td>
<td>4.207</td>
<td>4.238 (0.7)</td>
<td>4.266 (1.4)</td>
<td>4.276 (1.6)</td>
<td>4.394 (4.3)</td>
<td>4.477 (6.0)</td>
<td>4.603 (8.6)</td>
</tr>
<tr>
<td>Average</td>
<td>3.826</td>
<td>3.882 (1.4)</td>
<td>3.923 (2.5)</td>
<td>4.030 (5.1)</td>
<td>4.089 (6.4)</td>
<td>4.222 (9.4)</td>
<td>4.367 (12.4)</td>
</tr>
</tbody>
</table>
an emphasis on the performance of lossy coding, scalability, and other functional aspects, and although it is not necessarily appropriate to evaluate it alongside other methods that have been specialized for lossless coding, the lossless mode coding rate is provided as a reference value. Table 2 confirms that the proposed method (VS-AR-GG) was able to achieve encoding rates that were 6 to 44% better than the international standard method for lossless encoding, JPEG-LS. Furthermore, the proposed method exhibited the best coding rates for all of the images, and as far as the authors are aware, this is the highest level of performance obtained from a lossless encoder at the current time.

7. Summary

A high-compression lossless coding scheme for use with static images was investigated. The proposed scheme is based on the method in Ref. 8, which makes use of block-adaptive prediction, but with additional modifications to improve the coding rate. First, the probability density functions of the prediction errors were approximated by generalized Gaussian functions, which have a higher degree of freedom than the conventional Gaussian functions, and the efficiency with which the prediction errors are entropy coded was improved by using these functions as the probability models in the adaptive arithmetic encoding. A cost function was introduced that evaluates the sum of the code length of the prediction errors and the code length of the side information such that the total code length of an image is minimized by repeatedly optimizing each of the parameters that are needed during the encoding process based on this cost function. A variable-size block segmentation method that uses quadtrees was also investigated as a way to further improve the precision of the block-adaptive prediction. The compression ratios that were obtained as a result of including these modifications surpassed those of existing methods for all of the images used in the experiments, demonstrating the superior coding performance of the proposed method.

The proposed method features an extremely large difference between the number of operations during the encoding process and the decoding process because all of the computationally expensive coding parameter optimization processing is executed by the encoder. For example, the decoding time per image on the computer that was used in the experiments (Intel Xeon processor, 3.06 GHz, Linux OS) was less than 0.2 second for 512 x 512 pixel images, whereas the same images required 10 to 25 minutes to encode. Although the primary objective of this paper was to improve the encoding performance, the level of computation in the encoding process needs to be reduced for the method to become practically useful. To achieve this, although the design of the algorithm used in the optimization process is obviously important, it is also important that the parameters for the number of predictors (M), the prediction order (K), and the upper limit on the number of repetitions are set appropriately by considering the balance between encoder performance and computational load. In the future, we plan to examine the problem of extending the proposed method to lossless encoding of color images and video.

REFERENCES

10. Subbotin MT. On the law of frequency of error. Mat Shornik 1923;31:296–301.
13. Shirai N, Shiodera T, Matsuda I, Itoh S. Lossless coding using predictors and arithmetic code optimized iteratively for each image—Accuracy improvement in probability estimation of prediction


Authors (From left to right)

Ichiro Matsuda (member) received his B.S. (electrical engineering) and Ph.D. degrees from Tokyo University of Science in 1991 and 1996 and joined the Department of Electrical Engineering. He is now a lecturer in the Department of Electrical and Electronic Information. He received the Suzuki Memorial Scholarship from the Institute of Image Information and Television Engineers in 1994, the IEICE Young Researcher Encouragement Award in 2001, and the Best Paper Award from the Forum on Information Technology in 2002. He is currently conducting research into image coding. He is a member of the Institute of Image Information and Television Engineers.

Yuji Umezu (student member) received his B.S. (electrical engineering) from Tokyo University of Science in 2002 and is now in the Ph.D. program. He is currently conducting research into lossless image coding.

Nao Ozaki (student member) received his B.S. (electrical engineering) from Tokyo University of Science in 2002 and is now in the Ph.D. program. He is currently conducting research into lossless image coding.

Joji Maeda (member) received his B.S. (electronics) and Ph.D. degrees from the University of Tokyo in 1988 and 1993 and joined the Department of Electrical Engineering at Tokyo University of Science. He is now an assistant professor in the Department of Electrical and Electronic Information. He is a member of IEEE and the Optical Society of America.

Susumu Itoh (member) received his B.S. (electronics) and Ph.D. degrees from the University of Tokyo in 1976 and 1981 and joined the Department of Electrical Engineering at Tokyo University of Science. He is now a professor in the Department of Electrical and Electronic Information. He is a member of the Institute of Image Information and Television Engineers, the Information Processing Society of Japan, and the Institute of Electrical Engineers of Japan.