IEICE TRANS. FUNDAMENTALS, VOL. E80-A, NO. 11 NOVEMBER 1997

LETTER Special Section on Information Theory and Its Applications

A Bitplane Tree Weighting Method for Lossless Compression of Gray Scale Images*

Mitsuharu ARIMURA†, *Student Member*, Hirosuke YAMAMOTO†, *and* Suguru ARIMOTO††, *Members*

SUMMARY A Bitplane Tree Weighting (BTW) method with arithmetic coding is proposed for lossless coding of gray scale images, which are represented with multiple bitplanes. A bitplane tree, in the same way as the context tree in the CTW method, is used to derive a weighted coding probability distribution for arithmetic coding with the first order Markov model. It is shown that the proposed method can attain better compression ratio than known schemes with MDL criterion. Furthermore, the BTW method can be extended to a high order Markov model by combining the BTW with the CTW or with prediction. The performance of these modified methods is also evaluated. It is shown that they attain better compression ratio than the original BTW method without increasing memory size and coding time, and they can beat the lossless JPEG coding.

key words: *image compression, lossless compression, gray scale image, CTW method*

1. Introduction

When gray scale images are compressed by arithmetic coding, the gray scale data are usually treated as outputs from Markov sources. But, some problems occur since the alphabet size of such coding is usually large.

One of the problems is so-called "memory expansion." For example, the alphabet size of 8-bits gray scale is $2^8 = 256$. If these images are compressed with the first order Markov model, the size of states also becomes 256. Hence we need $256 \times 256 (= 65536)$ memory size to construct the Markov model. Furthermore, if we use a higher order of Markov model, this problem becomes more severe.

Another problem is so-called "over parameterization." Generally, image data can be represented more accurately as the number of Markov states becomes larger. However, since the size of image is limited, we cannot get enough samples for each state to estimate the probability distribution of the state when the number of states is large. For instance, the size of images is often $10^4 \sim 10^6$ pixels and it is not large enough for the first order or high order Markov model with 8 bit gray scale. These few samples worsen the compression ratio because of inaccurate estimate of probability distribution.

These problems can be overcome by treating some range of gray scale as one state [6]. In this paper, we reduce the number of states by using not all of 8 bitplanes but a few top bitplanes as a Markov state. Let x_i be the i-th pixel with 8 bits and let $S_d(x_i)$ be a state which consists of top d bits, $0 \leqq d \leqq 8$, of x_i. Then, in the case of the first order Markov model, we can calculate the coding probability of x_i as

$$P(x_i|S_d(x_{i-1})) = \frac{\alpha + n(x_i, S_d(x_{i-1}))}{256\alpha + n(S_d(x_{i-1}))}, \tag{1}$$

where $n(s)$ and $n(x, s)$ stand for the occurrence numbers of state s and pixel x in state s, respectively. α determines a prior probability of each x. Note that Eq. (1) represents the perfect first order Markov model for $d = 8$ while it represents the 0-th order Markov model, i.e., a memoryless model, for $d = 0$.

Since the number of states can be decreased by using this method with small d, the over parameterization problem can be reduced. However, if d becomes too small, $P(x_i|S_d(x_{i-1}))$ becomes to be independent of x_{i-1}. This worsens the compression ratio. Therefore, the optimal number d must be determined to attain high performance.

Zhang et al. [8] proposed a method which adaptively determines the optimal d, by the Minimum Description Length (MDL) criterion, that minimizes the criterion

$$L(d) = -\sum_{s \in S_d} \sum_{x \in \mathcal{A}} n(x, s) \log P(x|s)$$

at each i, where $\mathcal{A} = \{0, 1, \ldots, 255\}$ and $S_d = \{S_d(x); x \in \mathcal{A}\}$, i.e., a set of states which consist of top d bits. Their coding procedure with MDL criterion can asymptotically achieve the lower bound on the average redundancy shown by Rissanen [3], [4]. However, in the case of finite data sequence length, their method may not be optimal.

For coding of gray scale images, we propose a Bitplane Tree Weighting (BTW) method in this paper, which uses a weighting technique similar to the CTW method [7], instead of the MDL criterion. In the CTW

Manuscript received January 30, 1997.
Manuscript revised May 12, 1997.

†The authors are with the Faculty of Engineering, The University of Tokyo, Tokyo, 113 Japan.
††The author is with the College of Science and Engineering, Ritsumeikan University, Shiga-ken, 525-77 Japan.

*This work was presented in part at the 1996 International Symposium on Information Theory and Its Applications, Victoria, B.C., Canada.

method, a probability estimation of x_i is obtained by mixing up all $P(x_i|x_{i-1}\cdots x_{i-d+1})$, $0 \leqq d \leqq D$, where D is the maximum context length. This CTW method asymptotically achieves the same optimal performance that the MDL method does. However, it is known that, especially for short sequences, the performance of the CTW method is better than that of the MDL method. We apply the weighting technique of CTW method to the models with multiple bitplanes. Although the tree of the CTW method is constructed from the context of data, we construct a tree from bitplane data of pixels in our BTW method. The probability estimation of x_i is obtained by mixing up $P(x_i|S_d(x_{i-1}))$, $0 \leqq d \leqq 8$, in the same way as the CTW method.

The rest of the paper is organized as follows. In Sect. 2, we describes the algorithm of BTW method in detail. In Sect. 3, we compare the performance of the BTW method with that of two MDL methods, which use one path encoding based on the first order Markov model. We show that the BTW method can attain better performance than the MDL methods. Furthermore, the BTW method can be extended to a high order Markov model by combining the bitplane tree of the BTW method and the context tree of the CTW method. In Sect. 4, we consider such extension and evaluate its performance. We show that the compression ratio is improved by the extension without increasing memory size and coding time. In Sect. 5, we consider the BTW method with prediction. Finally, we show that these two modified methods can beat the lossless JPEG coding in the compression ratio.

2. BTW Algorithm

In this section, we describe a basic BTW method for the first order Markov model.

In the BTW method, the states of the first order Markov model are represented as nodes of a binary tree as shown in Fig. 1. We call this tree *a bitplane tree*. Each state corresponds to a gray scale range shadowed in the figure. The node of depth d corresponds to state $S_d(x_{i-1})$, which uses top d bitplanes of x_{i-1} as a state. The root state $S_0(x_{i-1}) = \lambda$ corresponds to the 0-th order Markov model, i.e., a memoryless model. Two

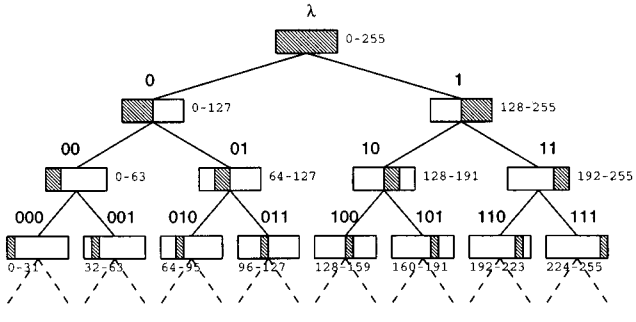

Fig. 1 Bitplane tree of simple Markov model.

nodes of depth 1 are distinguished by only the top bitplane of x_{i-1}, and these two nodes correspond to gray scale ranges $0 \sim 127$ and $128 \sim 255$, respectively. For $0 \leqq d \leqq 7$, state $S_d(x_{i-1}) = s$ has two children nodes $S_{d+1}(x_{i-1}) = s0$ and $S_{d+1}(x_{i-1}) = s1$. The nodes with $d = 8$ correspond to the perfect first order Markov model ($S_8(x_{i-1}) = x_{i-1}$).

The coding of BTW method is accomplished by recursively weighting the probabilities of nodes in the bitplane tree, in the same way as the CTW method. The estimated probability of sequence x^n on node s is calculated by

$$P_e^s(x^n) = \frac{\displaystyle\prod_{x:\, n(x,s)\geq 1} \prod_{i=0}^{n(x,s)-1} (\alpha + i)}{\displaystyle\prod_{j=0}^{n(s)-1} (256\alpha + j)}, \tag{2}$$

where α is a parameter which assigns a prior probability of x. In this paper, we use $\alpha = \frac{1}{2}$ which corresponds to the case of Jeffreys' prior. In case of $n(s) = 0$, i.e., state s has not appeared in x^n, we define $P_e^s(x^n)$ as $P_e^s(x^n) = 1$ instead of (2).

The weighted probability of sequence x^n is calculated in the same way as the CTW method for binary sequences [7] as follows.

$$P_w^s(x^n) = \begin{cases} \dfrac{P_e^s(x^n) + P_w^{s0}(x^n)P_w^{s1}(x^n)}{2} & \text{if } d(s) < 8 \\ P_e^s(x^n) & \text{if } d(s) = 8, \end{cases} \tag{3}$$

where $d(s)$ stands for the depth of node s[†]. Then

$$P(x_n|x^{n-1}) = \frac{P_w^\lambda(x^n)}{P_w^\lambda(x^{n-1})} \tag{4}$$

is used as the coding probability of gray scale x_n for adaptive arithmetic coding[††].

3. Comparison of BTW Method with MDL Methods

We now compare the performance of the BTW method with the MDL methods which are adaptive, i.e. one path encoding based on the first order Markov model. We consider the following two MDL methods.

MDL1 ([8])

Select d that minimizes

$$-\sum_{s\in S_d}\sum_{x\in\mathcal{A}} n(x,s)\log P(x|s).$$

[†] We can easily show that $P_w^s(x^n)$ gives consistently a probability distribution in the same way as [9].

[††] An arithmetic code for 256 symbols, which is shown in [2], is used.

Table 1 Comparison between BTW and MDL methods.

Images	Size (w × h) (pixels)	BTW with 8 bitplanes	BTW with 4 bitplanes	High order BTW	BTW with prediction	MDL1	MDL2	Lossless JPEG
aerial	256 × 256	6.148	6.169	6.026	5.991	6.579	6.460	6.038
camera		4.902	5.241	5.020	4.690	5.193	5.576	4.418
couple		4.379	4.733	4.468	4.281	4.767	4.624	4.616
girl		4.456	4.576	4.383	4.830	4.955	4.728	4.951
lady		5.245	5.421	5.199	4.836	5.685	5.474	4.932
moon		5.636	5.675	5.395	5.246	6.011	5.774	5.185
barbara	512 × 480	6.068	6.181	5.558	5.161	6.201	6.161	5.614
blackboard		4.410	4.569	4.279	4.042	4.536	4.509	4.458
boats		5.044	5.340	4.929	4.332	5.196	5.524	4.709
zelda		4.896	5.126	4.669	4.158	5.077	5.069	4.417
airplane	512 × 512	4.478	5.037	4.736	4.115	4.779	5.034	4.418
baboon		6.317	6.333	6.208	6.364	6.493	6.344	6.383
bridge		4.066	4.116	3.941	4.323	4.209	4.194	5.762
lenna		5.391	5.523	5.125	4.840	5.502	5.522	5.052
peppers		5.162	5.328	5.055	4.884	5.361	5.300	4.995
goldhill	720 × 576	5.035	5.274	4.957	4.772	5.171	5.211	4.993

(bits/pixel)

MDL2 ([3])

1. Let $d = 0$.
2. If

$$-\sum_{x\in\mathcal{A}} n(x, S_d(x_{i-1}))\log P(x|S_d(x_{i-1}))$$
$$\leqq -\sum_{x\in\mathcal{A}} n(x, S_d(x_{i-1})0)\log P(x|S_d(x_{i-1})0)$$
$$-\sum_{x\in\mathcal{A}} n(x, S_d(x_{i-1})1)\log P(x|S_d(x_{i-1})1),$$

   then use d.
3. Let $d = d + 1$. If $d = 8$, then use d. Otherwise, go to 2.

At each i, the algorithm MDL1 selects the optimal depth d while the algorithm MDL2 selects the optimal node $S_d(x_{i-1})$ of the bitplane tree. In both algorithms, pixel x_i is encoded using the distribution given by (1) with $S_d(x_{i-1})$.

In Table 1, the performance of the BTW method shown in the 3-rd column is compared with MDL methods shown in the 7-th and 8-th columns for sixteen standard images with 256-level gray scale. The BTW method compresses all images shorter than two MDL methods although we cannot say which of these two MDL methods is superior to the other.

In the 4-th column of Table 1, the performance of the BTW method is represented for the case that the maximum depth of d is restricted to 4 (top 4 bitplanes) rather than 8. In this case, the coding time becomes a half of 8 bitplanes case, and the memory size also decreases by this restriction. But, we note from the table that the degradation of performance is a little for many images.

4. BTW Method with a High Order Markov Model

In this section, we extend the BTW method to a high order Markov model by combining the bitplane tree of the BTW method with the context tree of the CTW method.

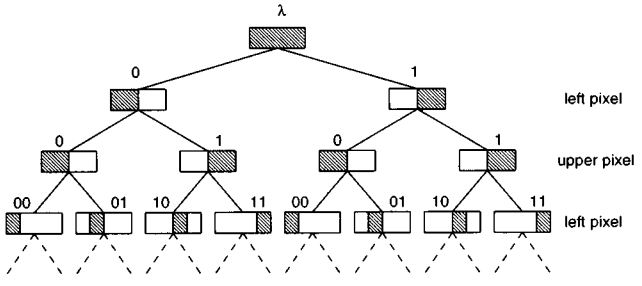

Fig. 2 Bitplane tree of a high order Markov model.

From the result stated in the last paragraph in the previous section, we note that the tree can be pruned to depth 4 in the first order Markov model with a little performance degradation. This pruned branches, i.e., saved memory, can be used for a high order Markov model. For instance, when we encode pixel x_i, we can use the top 4 bitplanes of the left pixel x_{i-1} and the upper pixel x_{i-L}, where L is the width of image, as a state of a high order Markov model. Although the bitplane trees of x_{i-1} and x_{i-L} can be combined in many ways, we combine them alternately from the top bitplane to the bottom bitplane of each x_{i-1} and x_{i-L} because they are supposed to be identically dependent on x_i.

Figure 2 shows the constructed bitplane tree of a high order Markov model, where the odd depths correspond to the left pixel x_{i-1} and the even depths correspond to the upper pixel x_{i-L}. For example, the nodes of depth 6 are related to the top 3 bitplanes of left pixel and the top 3 bitplanes of upper pixel. Using this combined bitplane, we can compress images by the same BTW algorithm described for the first order Markov model in Sect. 2.

In the 5-th column of Table 1, the performance of the BTW method with the high order Markov model

is also shown. From the table this modification improves the performance of BTW method for many images. It should be noted that since the total number of bitplanes are also 8 in the above high order Markov model, the performance is improved without increasing memory size and coding time.

Concerning images camera, couple, and airplane, this modification worsens the performance. The degradation comes from the fact that the performance of these images is relatively worse when the bitplane tree is pruned to depth 4 as shown in the 4-th column of the table. This degradation eats up the gain of the high order Markov model with two pixels. If we construct a bitplane tree of a high order Markov model by combining two or three bitplanes of three pixels, the performance degradation occurrs in all images because of the same reason.

5. BTW Method with Prediction

In this section, we present the BTW method with prediction. When we encode x_i, we can predict the value x_i from neighbor pixels. Hence, it is sufficient to encode the prediction error. We predict x_i from x_{i-L}, i.e., the upper pixel, since x_{i-1} is used in the BTW method. In order to restrict the range of the prediction error e_i, we use the following well-known representation [1].

$$e_i = \begin{cases} x_i - x_{i-L} + 256, & \text{if } x_i - x_{i-L} < 0, \\ x_i - x_{i-L}, & \text{otherwise.} \end{cases} \quad (5)$$

Then, e_i is encoded by the BTW with the first order Markov model. The bitplane tree is constructed based on e_{i-1}, i.e., the prediction error between x_{i-1} and x_{i-1-L}.

The performance of the BTW with prediction is shown in the 6-th column of Table 1. For the sake of comparison, Table 1 includes the performance of lossless JPEG coding [10], which is obtained by tuning its parameters for each image in order to achieve the maximum compression.

From Table 1, we can say that the BTW method with the high order Markov model is comparable with the lossless JPEG coding while the BTW method with prediction can beat the lossless JPEG coding. Especially, in the case of image bridge, the BTW method with the high order Markov model is about 1.8 bits/pixel better than the lossless JPEG coding.

As for the images girl and bridge, the performance of BTW method with the high order Markov model is better than that of BTW with prediction. We should note that the actual gray scale of image girl is 120 rather than 256. The performance degradation for this image occurs because the range of prediction error e_i, 215, is much larger than the actual gray scale, 120. Similarly, in the case of image bridge, the range of prediction error, 132, is much larger than the actual gray scale, 64.

The encoding speed of these BTW methods (about 1 Kbytes/second on SparcStation20) is much slower than the lossless JPEG coding (about 100 Kbytes/second). But the speed of BTW methods may be improved since our program of the BTW method is not tuned up for speed.

6. Conclusion

In this paper, a Bitplane Tree Weighting method is proposed for gray scale images. Our results show that the BTW method can attain better compression than the known schemes with MDL criterion in the first order Markov model.

Furthermore, we modified the BTW algorithm in two ways. The one is the combination of the bitplane tree in the BTW method and the context tree in the CTW method. The other modification is the BTW method with prediction. We showed that these modification can improve the performance without waste of memory size and encoding time, and they can beat the lossless JPEG coding.

In the methods described in this paper, we constructed the bitplane tree based on x_{i-1} and/or x_{i-L}. However, we can consider another kind of bitplane tree based on x_i itself as proposed by Shiki et al. [5]. Combining the BTW method with their method, we may further improve the performance.

References

[1] S. Kitada, H. Morita, and S. Arimoto, "Dynamically adaptive coding of digitized gray-level images," IEICE Trans., vol.J68-D, no.4, pp.639–646, April 1985.

[2] M. Nelson and J.L. Gailly, "The data compression book," second edition, M&T Books, 1995.

[3] J. Rissanen, "Universal coding, information, prediction, and estimation," IEEE Trans. Inf. Theory, vol.IT-30, no.4, pp.629–636, July 1984.

[4] J. Rissanen, "Complexity of strings in the class of markov sources," IEEE Trans. Inf. Theory, vol.IT-32, no.4, pp.526–532, July 1986.

[5] J. Shiki, S. Itoh, and T. Hashimoto, "A CTW-like coding method for sources with large alphabet size," Proc. 17th Symposium on Inf. Theory and Its Appl. (SITA '94), Hiroshima, Japan, pp.715–718, Dec. 1994.

[6] S. Todd, G.G. Langdon, Jr., and J. Rissanen, "Parameter reduction and context selection for compression of gray-scale images," IBM J. Res. Develop., vol.29, no.2, pp.188–193, March 1985.

[7] F.M.J. Willems, Y.M. Shtarkov, and T.J. Tjalkens, "The context tree weighting method: Basic properties," IEEE Trans. Inf. Theory, vol.41, no.3, pp.653–664, May 1995.

[8] Q. Zhang, R. Kohno, and H. Imai, "A scheme of arithmetic coding for gray scale images with MDL criterion," IEICE Trans., vol.J77-A, no.8, pp.1157–1166, Aug. 1994.

[9] N. Yokoo and T. Kawabata, "A simple implementation of context tree weighting method and its verification," IEICE Technical Report, IT93-123, March 1994.

[10] Portable Video Research Group, "PVRG-JPEG CODEC version 1.2," Stanford University, 1994.