Improved Image Compression Using S-Tree and Shading Approach

Kuo-Liang Chung and Jung-Gen Wu, *Member, IEEE*

***Abstract*—Recently, Distasi *et al.* presented a storage-saving image compression method called B-tree triangular coding (BTTC) method. Based on the modified S-tree data structure and the Gouraud shading method, this paper presents an improved image compression method called the S-Tree Compression (STC) method. Experimental results illustrate that the bit rates and the image quality of the proposed STC method are quite competitive with the BTTC method. Due to the simple geometrical decomposition in the STC method, the ratio of the execution time of the proposed method over the BTTC method is less than 1/2.**

***Index Terms*—Gouraud shading, image compression, image quality, S-Tree.**

I. Introduction

THERE are many image compression techniques [8], [4], [11], [5] that have been developed. Although the compression ratio is very important, however, sometimes the encoding and decoding time, i.e., the execution time, is critical in the applications of real-time image retrieval and communication.

Recently, Distasi *et al.* [2] presented an efficient B-tree triangular coding (BTTC) method for compressing gray images. The encoding can be performed in $O(n \log n)$ time; the decoding can be performed in $O(n)$ time, where n denotes the number of pixels in the image. The experimental results illustrate that the BTTC method produces images with satisfactory quality and has shorter execution time when compared to the JPEG method, although the bit rates are higher by a factor of about 2. In [2], for each partitioned block in the image domain, a triangle is used to represent that block and it needs 8 multiplications, 2 divisions, and 12 additions to estimate the greylevel of the pixel at position (x, y) within the block. This estimation is used frequently in both encoding phase and the decoding phase, so it dominates the execution time.

Based on the modified S-tree data structure and the Gouraud shading method [3], this paper presents a modified image compression method called the S-Tree Compression (STC) method. Like the results in [2], the encoding in the STC method takes $O(n \log n)$ time; the decoding takes $O(n)$ time. Due to the low-cost computation of the Gouraud shading and the simple regular structure of S-tree, transforming the given gray image into the S-tree representation can be implemented efficiently. Experimental results show that the proposed STC method has a superior performance in the execution time (less than half of the BTTC method) without sacrificing compression ratio and image quality. Considering one noise allowable in each block, the proposed STC method can be modified slightly to achieve a dramatic compression improvement and speed up the execution time while still preserving a satisfactory image quality.

The rest of this paper is organized as follows. Section II presents the proposed STC method for compressing gray images. In Section III, some experiments on a real image are carried out to demonstrate and compare the performance of the proposed method and the BTTC method. The case of one noise allowable in one block is discussed in the same section. Some concluding remarks are addressed in Section IV.

II. Proposed STC Method

The proposed STC method consists of two phases. In the encoding phase, the given image is partitioned into some homogeneous blocks based on the bintree, i.e., binary tree, decomposition principle, then the S-tree representation is used to represent these blocks. The reason that we do not adopt the quadtree encoding is that Shaffer *et al.* [10] have shown that the bintree encoding is more efficient and simpler than the quadtree storage. In empirical comparisons, the bintree encoding has about 25% space utilization improvement over quadtree encoding. Next, a level-compression technique, cutting off the head and the tail of the string representation [9], [2] is employed to represent the corresponding S-tree in a more compact way. In the encoding phase and the decoding phase, the Gouraud shading method is used to control the image quality under a specified error tolerance.

According to the depth-first search (DFS) technique, the bintree representation is based on the recursive subdivision of the image into two equal-sized subimages. At each step, the partition is alternated between the x- and y-axes. If the subimage is not a homogeneous block, it is subdivided into two equal-sized subimages until all homogeneous blocks are obtained. The formal definition of a homogeneous block is given in the following paragraph.

During subdividing the image, a block is called a homogeneous block if the estimated greylevel of each of the pixels in the block is in some vicinity of its real greylevel. We know that there are four corners in one block. Suppose the coordinates of the four corners are (x_1, y_1), (x_2, y_1), (x_1, y_2), and (x_2, y_2) with greylevels g_1, g_2, g_3, and g_4, respectively. Using the Gouraud shading method [3], the estimated greylevel of the pixel at (x, y) in the block is calculated as

$$g_{est}(x, y) = g_5 + \frac{g_6 - g_5}{y_2 - y_1}(y - y_1)$$

Paper approved by K.-K. Ma, the Editor for Video and Signal Processing of the IEEE Communications Society. Manuscript received December 27, 1998; revised June 8, 1999 and October 18, 1999. The work of K.-L. Chung was supported by NSC88-2213-E011-006. The work of J.-G. Wu was supported by NSC89-2614-H003-001-F20.

K.-L. Chung is with the Department of Information Management, Institute of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan 10672, R.O.C. (e-mail: klchung@cs.ntust.edu.tw).

J.-G. Wu is with the Department of Information and Computer Education, National Taiwan Normal University, Taipei, Taiwan 10610, R.O.C. (e-mail: jgwu@ice.ntnu.edu.tw).

Publisher Item Identifier S 0090-6778(00)03999-4.

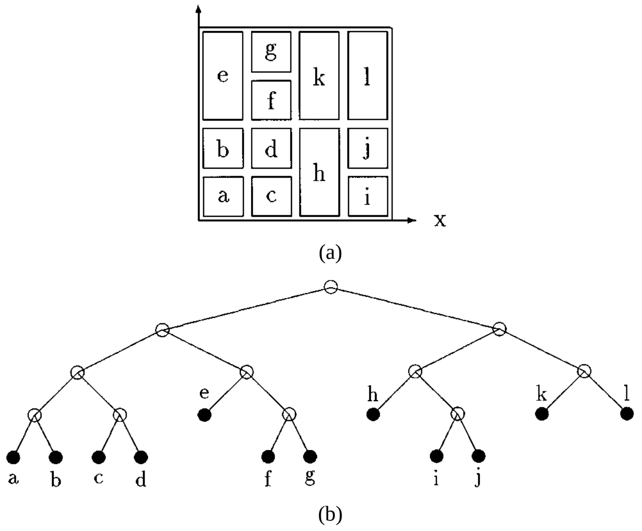

Fig. 1. One example. (a) Homogeneous blocks. (b) The bintree representation.

where

$$g_5 = g_1 + \frac{g_2 - g_1}{x_2 - x_1}(x - x_1)$$

and

$$g_6 = g_3 + \frac{g_4 - g_3}{x_2 - x_1}(x - x_1). \tag{1}$$

Given a specified error tolerance ε, if the following image quality condition holds:

$$|g(x, y) - g_{\text{est}}(x, y)| \le \varepsilon \tag{2}$$

for all the estimated pixels at position (x, y)'s in the block, $x_1 \le x \le x_2$ and $y_1 \le y \le y_2$, then the block is called a homogeneous block.

For calculating the above estimated greylevel of one pixel at (x, y) [see (1)], it takes two divisions, three multiplications, and ten additions, where we assume that the time required to perform one addition is equal to one subtraction. Under the same error tolerance [see (2)], in [2], it takes 8 multiplications, 2 divisions, and 12 additions to estimate the greylevel of the pixel at position (x, y). For one estimated greylevel, the time required in our adopted method is much less than that in [2]. Since there are a large amount of estimated greylevels to be calculated in the encoding phase and the decoding phase, the proposed STC method does have the computational advantage. In fact, in each partition in the bintree decomposition scheme, due to the simple vertical or horizontal splitting it takes much less time when compared to that required in the triangulation decomposition scheme. This is another computational advantage of the proposed method.

For exposition, suppose one gray image has been partitioned into some homogeneous blocks, as shown in Fig. 1(a), according to the bintree decomposition scheme. The corresponding bintree is illustrated in Fig. 1(b).

The original S-tree representation [6] for representing the bintree is based on the DFS technique. The modified S-tree is based on the breadth-first search (BFS) technique [1] and consists of two tables, the linear-tree table, and the color table. Due to the BFS manner, a level-compression technique mentioned above can be employed to reduce the memory requirement in the modified S-tree representation.

Traversing the bintree in a BFS manner, at each time, we emit a "0" when an internal node is encountered; emit a "1" when a leaf node is encountered. After traversing the bintree, the sequence of these ordered binary values is saved in the linear-tree table. Meanwhile, at each time, we do nothing when an internal node is encountered. When a leaf node is encountered, we emit the four greylevels of the four corners in the related homogeneous block, say $(a_{ul}, a_{ur}, a_{bl}, a_{br})$, for the greylevels of the upper-left, upper-right, bottom-left, and bottom-right corners of the homogeneous block a. The sequence of these ordered values is stored in the color table. The modified S-tree representation for Fig. 1(b) is listed below

linear-tree table:	0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 1 1 1 1 1 1 1
color table:	$(e_{ul}, e_{ur}, e_{bl}, e_{br}), (h_{ul}, h_{ur}, h_{bl}, h_{br}), \cdots, (j_{ul}, j_{ur}, j_{bl}, j_{br})$.

Observing the bintree in Fig. 1(b), the top-most three levels form a complete subtree. So, the nodes at the top-most two levels are all internal nodes and the corresponding entries in the linear-tree table for these internal nodes are all 0's. To reduce the memory required in the modified S-tree representation, we discard the leading seven 0's, which are corresponding to the root node and the nodes at the top-most two levels, and only store the value 2 to indicate the discarded depth. This leads to a memory-saving effect. It is also observed that the eight nodes at the bottom level are all leaves. We discard the eight "1"s to reduce the data structure. Now, the reduced linear-tree table becomes 0 0 1 0 1 0 1 1.

Return to the bintree decomposition scheme. We propose two subdividing methods, STC1 and STC2. STC1 subdivides the image into nonoverlapping homogeneous blocks. For example, the original $2^n \times 2^n$ image is subdivided into two $2^n \times 2^{n-1}$ nonoverlapping blocks. The greylevels of the four corners of each homogeneous block are stored. STC2 subdivides the image into overlapping homogeneous blocks with shared edges. An augmented image with size $(2^n + 1) \times (2^n + 1)$ is obtained by duplicating the last column and last row of the original image. Each homogeneous block is of size $(2^k + 1) \times (2^k + 1)$ or $(2^k + 1) \times (2^{k-1} + 1)$ for some $k \le n$. For example, in Fig. 1, the homogeneous block a shares its top edge with the bottom edge of the homogeneous block b; that is, $a_{ul} = b_{bl}$ and $a_{ur} = b_{br}$. The shared data need to be stored only once. Although it is easier to implement and manipulate the data structure of the first method, the experiments show that the first method needs more storage space to save the corresponding data structure.

III. Experimental Results

We use the 512 × 512 Lena image to compare the performance of the proposed STC1 method, STC2 method, and the previous BTTC method. The experiments are performed on the IBM compatible personal computer Pentium II microprocessor with 233 MHz.

The image used for BTTC and STC2 is of size 513 × 513 and is generated by duplicating the last column and the last row of the original image. Although it is shown that the number of blocks in STC1 is much less than that in STC2, the memory

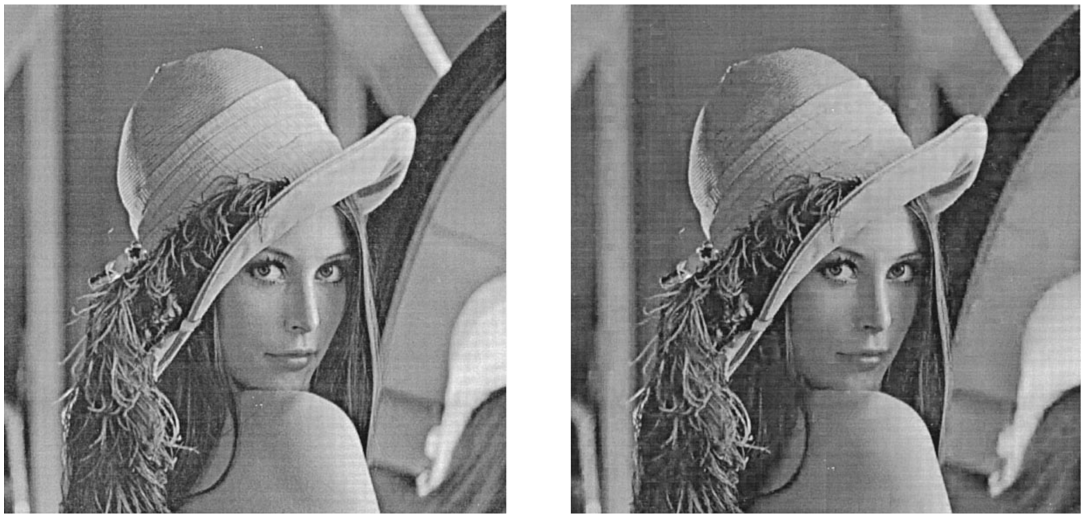

Fig. 2. (a) The original and (b) the result for error tolerance $\varepsilon = 21$.

TABLE I
COMPRESSION STATISTICS

	number of blocks			number of minimal blocks			Bits per pixel		
ε	BTTC	STC1	STC2	BTTC	STC1	STC2	BTTC	STC1	STC2
5	290567	54686	173822	163288	47290	129206	6.40	6.90	7.36
9	146627	33452	87010	58002	22424	49352	3.47	4.25	3.86
13	88085	21790	51895	28334	13060	25832	2.13	2.78	2.33
17	62835	16343	36361	16544	8834	16104	1.55	2.09	1.65
21	47448	13094	27305	10008	6416	10534	1.19	1.67	1.26

requirement in STC2 is still less than that in STC1 due to the memory-saving advantage of sharing edges. The blocks found in BTTC also share edges.

Given five kinds of error tolerances, $\varepsilon = 5, 9, 13, 17,$ and 21, Table I lists the number of blocks, the number of minimal blocks, and the average bits per pixel. A minimal block contains the minimal number of pixels allowable, i.e., four pixels for STC's and three pixels for BTTC. No data compression is achieved in encoding a minimum block. It is observed in Table I that the average bits per pixel of STC2 is competitive with that of BTTC. Fig. 2 shows the original image and the result of STC1 for one error tolerance. In fact, it is difficult to distinguish the difference from the two images according to our visual system. Fig. 3 illustrates the partitioned blocks of the image by STC1.

Table II lists the signal to noise ratios (SNR's) for the decoded images, the encoding time, and the decoding time. The SNR for a decoded image is calculated as

$$\mathrm{SNR(dB)} = 10 \log_{10} \frac{\sum_{x=1}^{m}\sum_{y=1}^{m} g^2(x, y)}{\sum_{x=1}^{m}\sum_{y=1}^{m} (g(x,y) - g_{est}(x, y))^2}. \quad (3)$$

It is observed that under the same error tolerance, STC2 and BTTC have the similar SNR [see (3)]; STC1 has a little more SNR when compared to BTTC by sacrificing the average bits per pixel. The time unit for the encoding time and the decoding time is "second." It is observed that the ratio of the execution time (including the encoding time and the decoding time) of STC1 or STC2 over BTTC is less than 1/2.

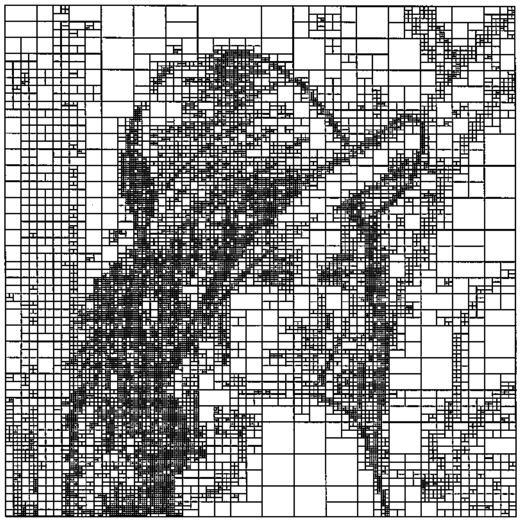

Fig. 3. The partitioned blocks using the STC1 method.

TABLE II
SNR, ENCODING TIME, AND DECODING TIME

	SNR			Encoding time			Decoding time		
ε	BTTC	STC1	STC2	BTTC	STC1	STC2	BTTC	STC1	STC2
5	36.6	41.0	37.9	8.49	1.56	4.71	6.46	1.15	3.24
9	30.6	32.6	31.0	4.89	1.12	2.61	3.54	0.76	1.69
13	27.9	29.3	28.2	3.46	0.85	1.74	2.30	0.56	1.09
17	26.1	27.2	26.2	2.59	0.68	1.30	1.76	0.46	0.82
21	24.6	25.6	24.7	2.24	0.59	1.04	1.43	0.40	0.66

The results about the SNR and bits per pixel (BPP) shown in Table I do not agree with those in [2]. The causes of this disagreement are explained below as the following.

1) The linear interpolation [see (1)] used for estimating the greylevel of a pixel generates a real number, but the final result is truncated to an integer. Besides, we simulated the algorithm stated in [2], but may have a different sequence for the arithmetic operations. This may produce some differences.

TABLE III
MEMORY REQUIREMENTS FOR THE MODIFIED METHODS

	number of blocks		number of minimal blocks		Bits per pixel	
ε	STC1A	STC2A	STC1A	STC2A	STC1A	STC2A
5	43924	109177	29824	43698	5.58	4.81
9	24946	53220	13178	16540	3.18	2.40
13	16959	33808	8248	9603	2.17	1.54
17	13109	24599	5676	5800	1.68	1.14
21	10545	18906	3958	3680	1.35	0.88

TABLE IV
SNR, ENCODING TIME, AND DECODING TIME FOR THE MODIFIED METHODS

	SNR		Encoding time		Decoding time	
ε	STC1A	STC2A	STC1A	STC2A	STC1A	STC2A
5	35.1	30.9	1.37	3.30	0.95	2.11
9	30.1	27.9	0.94	1.80	0.62	1.12
13	27.6	26.1	0.71	1.29	0.47	0.77
17	25.9	24.7	0.62	1.03	0.40	0.60
21	24.3	23.4	0.53	0.86	0.35	0.50

2) The greylevels stored in the BTTC method are encoded using Huffman coding method. So the BPP's shown in [2] are much less than those shown in Table I.

It is also observed that the number of minimum blocks is about $1/2$ of that of the total blocks. No data compression is achieved in encoding these minimum blocks. In order to get a better compression ratio, we allow one pixel in a homogeneous block to exceed the specified error tolerance. The resulting memory requirement, SNR's, encoding time, and decoding time needed in the modified methods, namely, STC1A and STC2A (with respect to STC1 and STC2), are listed in Tables III and IV, respectively. It is shown that dramatic improvements in both compression ratio and execution time have been achieved while still preserving a satisfactory image quality. Fig. 4 illustrates the result of STC1A for the case of error tolerance $\varepsilon = 21$. In fact, it is still difficult to distinguish the difference between the image and the original image according to our visual system.

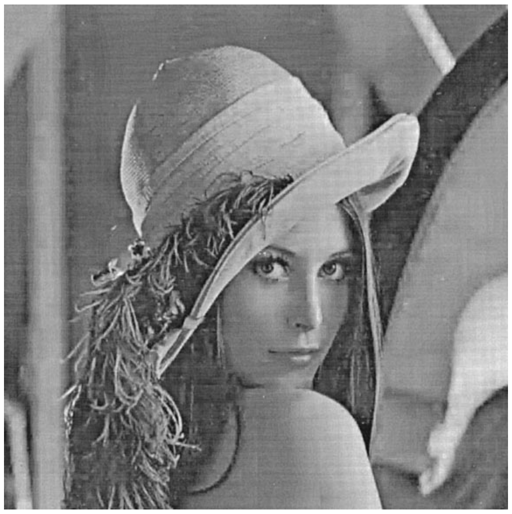
Fig. 4. The result for error tolerance $\varepsilon = 21$ when allowing one noise in each block.

IV. CONCLUSIONS

We have presented some variants of a novel method for compressing gray images. Experimental results illustrate that the bit rates and the image quality of the proposed STC method are quite competitive with the BTTC method, but the ratio of the execution time of the proposed method over the BTTC method is less than $1/2$. Considering one noise allowable in each block, the variants of proposed STC method can achieve a dramatic compression improvement and speed up the execution time while still preserving a satisfactory image quality.

Previously, Jordan *et al.* [7] presented a polygonal approximation approach for progressive content-based shape compression for retrieval of binary images. For different levels of given errors tolerance, it seems that the proposed modified S-tree representations in this paper can be used in the application of progressive transmission for gray images, and the extension of [7] is our future research topic.

ACKNOWLEDGMENT

The authors would like to thank the two reviewers, Dr. K.-K. Ma and Dr. W.-N. Yang, for making many valuable suggestions and corrections that led to the improved version of the paper.

REFERENCES

[1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 1990.

[2] R. Distasi, M. Nappi, and S. Vitulano, "Image compression by B-tree triangular coding," *IEEE Trans. Commun.*, vol. 45, pp. 1095–1100, Sept. 1997.

[3] J. D. Foley, A. V. Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics, Principle, and Practice*, 2nd ed. Reading, MA: Addison-Wesley, 1990.

[4] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Boston, MA: Kluwer Academic, 1992.

[5] G. Held and R. Marshall, *Data and Image Compression: Tools and Techniques*. New York: Wiley, 1996.

[6] W. D. Jonge, P. Scheuermann, and A. Schijf, "S^+-Trees: An efficient structure for the representation of large pictures," *Comput. Vision Image Understanding*, vol. 59, pp. 265–280, 1994.

[7] C. L. B. Jordan, T. Ebrahimi, and M. Kunt, "Progressive content-based shape compression for retrieval of binary images," *Comput. Vision Image Understanding*, vol. 71, no. 2, pp. 198–212, 1998.

[8] A. N. Netravali and B. G. Haskel, *Digital Pictures—Representation and Compression*. New York: Plenum, 1988.

[9] H. Samet, *The Design and Analysis of Spatial Data Structures*. New York: Addison-Wesley, 1990.

[10] C. A. Shaffer, R. Juvvadi, and L. S. Health, "Generalized comparison of quadtree and bintree storage requirements," *Image Vision Computing*, vol. 11, no. 7, pp. 402–412, 1993.

[11] W. B. Pennebaker and J. L. Mitchell, *JPEG: Still Image Data Compression Standard*. New York, NY: Van Nostrand Reinhold, 1993.