Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

PERGAMON

Pattern Recognition 37 (2004) 1591–1605

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/patcog

Efficient region segmentation on compressed gray images using quadtree and shading representation☆

Kuo-Liang Chung[a,*], Hsu-Lien Huang[a], Hsueh-I Lu[b]

[a] *Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, No. 43, Section 4, Keelung Road, Taipei 10672, Taiwan, ROC*
[b] *Institute of Information Science, Academia Sinica, No. 128, Section 2, Academia Road, Taipei 115, Taiwan, ROC*

Received 6 August 2003; received in revised form 23 February 2004; accepted 23 February 2004

Abstract

Image segmentation, which partitions the image into homogeneous regions, is a fundamental operation in image processing. Suppose the input gray image with size $N \times N$ has been compressed into the compressed image via quadtree and shading representation. Assume that the number of blocks in the representation is B, commonly $B < N^2$ due to the compression effect. This paper first derives some closed forms for computing the mean/variance of any block and for calculating the two statistical measures of any merged region in $O(1)$ time. It then presents an efficient $O(B\alpha(B))$-time algorithm for performing region segmentation on the compressed image directly where $\alpha(B)$ is the inverse of the Ackerman's function and is a very slowly growing function. With the same time complexity, our results extend the pioneering results by Dillencourt and Samet from the map image to the gray image. In addition, with four real images, experimental results show that our proposed algorithm has about 55.4% execution time improvement ratio when compared to the previous fastest region-segmentation algorithm by Fiorio and Gustedt whose $O(N^2)$-time algorithm is run on the original $N \times N$ gray image.

Keywords: Gouraud shading; Image compression; Quadtree; Region growing; Region segmentation; Union-find

1. Introduction

In image processing [1,2], given a gray image, it is important to segment the image into homogeneous regions such that each region has special content, e.g., the distribution of gray levels in each region is under some allowable statistical measures. This segmentation work is also called the region segmentation. After that, these segmented regions can be used to determine the objects in which usually each object consists of some consecutive regions. These determined objects find useful applications in image retrieval [3].

There are several techniques to perform the image segmentation [1,2,4,5], ranging from the simple approach to the complex approach. Different kinds of image segmentation methods lead to different segmented images. For example, in thresholding method, we use the obtained thresholds, say k thresholds, to transform the input gray image into the segmented image with $k + 1$ gray levels. However, using the thresholding approach, the segmented image lacks the characteristic of separated regions. Region growing approach relies on aggregating the neighboring pixels based on some growing principles, and finally the input image is transformed into some segmented regions. In this research, we adopt the region growing approach for performing the image segmentation. Previously, considering a map image, Dillencourt and Samet [6] presented a pioneering region-segmentation algorithm on the quadtree representation [7], where each leaf node represents a block with constant color. On the gray image of size $N \times N$, Fiorio and

☆ Supported by the National Science Council of ROC under contracts NSC90-2213-E011-056 and NSC92-2213-E011-080.
* Corresponding author. Tel.: +886-2273-76991; fax: +886-2273-76777.
E-mail address: klchung@cs.ntust.edu.tw (K.-L. Chung).

0031-3203/$30.00
doi:10.1016/j.patcog.2004.02.009

Gustedt [8] presented two linear-time, i.e., $O(N^2)$-time, algorithms for performing the region segmentation. The special version of the union-find strategy used in Ref. [8] further leads to other applications [9].

Previously, based on the B-tree triangular approach, Distasi et al. [10] presented an efficient image compression method. Their method has shorter execution time than that of the standard JPEG [11], although the bit rate is higher by a factor of about 2. Based on the S-tree data structure [12] and the Gouraud shading technique [13], an improved compression method called the STC method [14] is presented to partition the given image into a set of blocks. Under the same compression ratio, the STC method has shorter encoding/decoding time than that in Ref. [10] while preserving the same image quality. In fact, the STC method can be viewed as a promising spatial data structure (SDS) that extends the previous SDSs [7] from the binary image domain to the gray image domain.

Instead of using the binary tree decomposition principle in Ref. [14], we adopt the quadtree decomposition and suppose the gray image has been compressed into the compressed image using the quadtree and shading representation where the number of generated blocks is B and each block is not with constant color, commonly $B < N^2$ due to the compression effect. This paper first derives some closed forms for computing the mean/variance of any block and for calculating the two statistical measures of any merged region in $O(1)$ time. Then, this paper presents an efficient $O(B\alpha(B))$-time algorithm for performing region segmentation on the compressed image directly where $\alpha(B)$ is the inverse of the Ackerman's function and is a very slowly growing function. The time complexity of our proposed algorithm is nearly linear. With the same time complexity, the results of this paper extend the previous pioneering results by Dillencourt and Samet [6] from the map domain to the gray image domain. In addition, with four real images, experimental results show that our proposed algorithm has about 55.4% execution time improvement ratio when compared to the previous fastest algorithm by Fiorio and Gustedt [8] whose algorithm is run on the original $N \times N$ gray image.

2. The QS-based compressed images

Based on the compression method [14], this section presents a quadtree—and shading-based (QS-based) compression method. For convenience, this method is called the QSC method. In the QSC method, the original image is first partitioned into homogeneous blocks based on the quadtree decomposition principle. During the partition processing, the Gouraud shading method [13] is used to control the image quality under a specified error tolerance.

The quadtree decomposition is based on recursively dividing the image into four equal-sized subimages, i.e., quadrants, and they are labeled *nw* (northwest), *ne* (northeast), *sw* (southwest), and *se* (southeast), respectively. If a subimage is not a homogeneous block, it is subdivided into four equal-sized subimages again until all the blocks are homogeneous. A block is called a homogeneous block if the estimated gray levels of this block are in some vicinity of its real gray levels. In what follows, the formal definition of a homogeneous block is given.

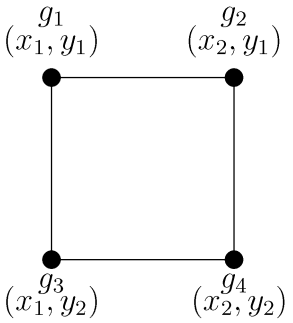

Fig. 1. A homogeneous block.

As shown in Fig. 1, there are four corners for a homogeneous 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); their gray levels are denoted by g_1, g_2, g_3, and g_4, respectively. Using the Gouraud shading method, the estimated gray level g_{est} of the pixel at (x, y) in the block is calculated by

$$g_{est}(x, y) = g_5 + (g_6 - g_5) \times r_1, \tag{1}$$

where $g_5 = g_1 + (g_2 - g_1) \times r_2$, $g_6 = g_3 + (g_4 - g_3) \times r_2$, $r_1 = (y - y_1)/(y_2 - y_1)$, and $r_2 = (x - x_1)/(x_2 - x_1)$.

Specifically, g_5 (g_6) is the estimated gray level of the pixel at (x, y_1) ((x, y_2)). Given a specified error tolerance ε, if the image quality $|g(x, y) - g_{est}(x, y)| \leqslant \varepsilon$ holds for all the pixels in the block, $x_1 \leqslant x \leqslant x_2$ and $y_1 \leqslant y \leqslant y_2$, then we say that this block is homogeneous. In the estimated block, the gray levels are smoothly varied within the four corners.

For example, one gray image has been partitioned into some homogeneous blocks as shown in Fig. 2(a) according to the above partition principle. The corresponding quadtree is illustrated in Fig. 2(b). Each homogeneous block in Fig. 2(a) corresponds to a leaf node of the quadtree in Fig. 2(b).

Now two tables are used to save the quadtree structure and the related gray levels efficiently. One table is the linear-tree table which is used to record the quadtree structure and the other is the color table which is used to record the related gray levels on the corners of these homogeneous blocks. First, we traverse the quadtree in a depth first search (DFS) manner, when an internal node of the quadtree is encountered, we add a '0' to the linear-tree table; when a leaf of the quadtree is encountered, we add a '1' to the linear-tree table.

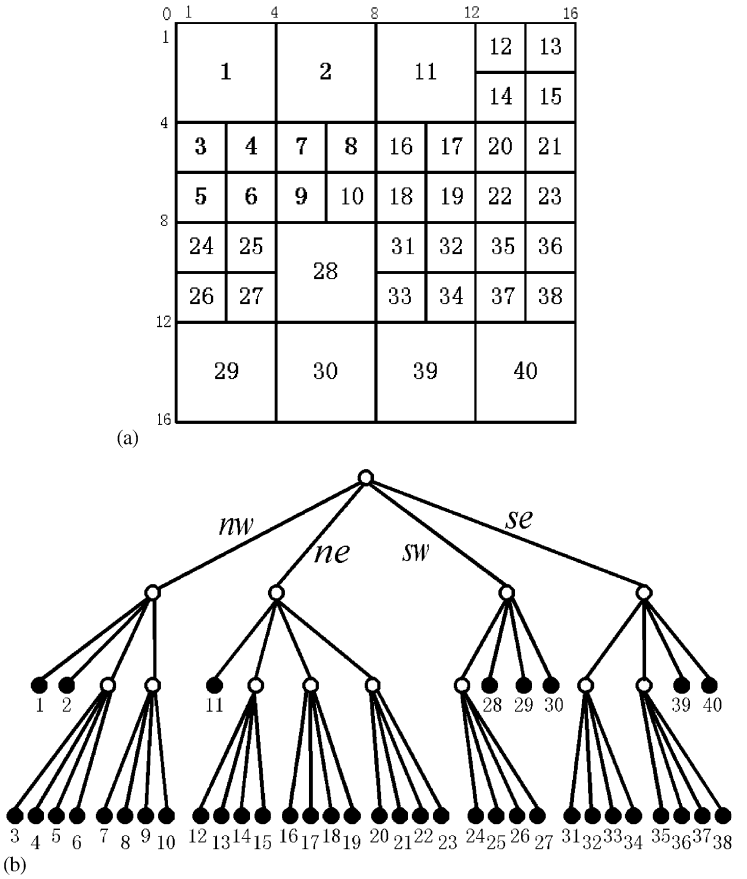

Fig. 2. (a) An example and (b) the quadtree representation.

After traversing the quadtree in a DFS manner, we can get a sequence of ordered binary values of the linear-tree table. In fact, besides representing the quadtree structure, the linear-tree table can be used to guide the sweep of the whole image in our region-segmentation algorithm. During traversing the quadtree, each time a leaf node is encountered, we add a four-tuple entry to the color table. The values in the four-tuple entry added for a leaf of the quadtree are the four corners' gray levels, namely I_{ul}, I_{ur}, I_{bl}, and I_{br}, where I_{ul}, I_{ur}, I_{bl}, and I_{br} denote the gray levels of the upper-left corner, upper-right, bottom-left, and bottom-right, respectively, of the block indexed with I. After traversing the quadtree in a DFS manner, a sequence of ordered four-tuple entries are stored in the color table. The contents of the linear-tree table and the color table for Fig. 2(a) are listed below:

linear-tree table : 00110111101111010111101111 0

11110011111110011110111111,

color table : $(1_{ul}, 1_{ur}, 1_{bl}, 1_{br}), (2_{ul}, 2_{ur}, 2_{bl}, 2_{br})$,

$\ldots, (40_{ul}, 40_{ur}, 40_{bl}, 40_{br})$.

The data structure consisting of the above two tables is also called the S-tree representation. When traversing the linear-tree table, if the bit '1' is found, we jump to the color table and can know the size and the four corners' gray levels of the scanned leaf node, i.e., block.

On the above S-tree representation, for a given error tolerance ε, suppose the linear-tree table has ℓ bits where the number of 1's is k_1, then the number of four-tuple entries in the color table is also equal to k_1.

3. Our region-segmentation algorithm

This section presents the main contributions of the paper. First, we derive some closed forms for computing the mean/variance of any homogeneous block and for calculating the mean/variance of a newly merged region in $O(1)$ time. Secondly, based on the data structures used in Ref. [6], we present a modified data structure which will be used in our region-segmentation algorithm on the above S-tree representation. Then a simulation is given. Finally, the formal algorithm is given.

3.1. Computing the mean/variance of a block in $O(1)$ time

Before presenting our proposed region-segmentation algorithm on the S-tree representation, we first present an efficient method for calculating the mean and the variance of a block in $O(1)$ time. The calculated mean and the variance of one block will be used in the region-merging process.

Consider the top boundary of Fig. 1, i.e., the segment between (x_1, y_1) and (x_2, y_1). For convenience, let this segment be denoted by s_t. Similarly, the segment between (x_1, y_2) and (x_2, y_2) is denoted by s_b; the segment from (x_1, y_1) to (x_1, y_2) is denoted by s_l and the segment from (x_2, y_1) to (x_2, y_2) is denoted by s_r. It is clear that the estimated gray level changes per displacement along s_t, s_b, s_l, and s_r, respectively, are equal to

$$\Delta g_{s_t} = \frac{g_2 - g_1}{x_2 - x_1} = \frac{g_2 - g_1}{n-1},$$

$$\Delta g_{s_b} = \frac{g_4 - g_3}{x_2 - x_1} = \frac{g_4 - g_3}{n-1},$$

$$\Delta g_{s_l} = \frac{g_3 - g_1}{y_2 - y_1} = \frac{g_3 - g_1}{n-1}$$

and

$$\Delta g_{s_r} = \frac{g_4 - g_2}{y_2 - y_1} = \frac{g_4 - g_2}{n-1}, \tag{2}$$

where n denotes the length of the side of the homogeneous block.

From Eq. (2), the estimated gray level at position (x_1, y_1+1) is $g_{est}(x_1, y_1+1) = g_1 + \Delta g_{s_l}$. Further, we want to compute the difference of gray level from (x_1+1, y_1) to (x_1+1, y_1+1). Equivalently, we want to compute $\Delta g_{y_1} = g_{est}(x_1+1, y_1+1) - g_{est}(x_1+1, y_1)$. We consider the segment s_{l+1} between (x_1+1, y_1) and (x_1+1, y_2). The estimated gray level change per displacement along the segment s_{l+1} is equal to $((g_3 + \Delta g_{s_b}) - (g_1 + \Delta g_{s_t}))/(n-1)$. In fact, the difference of gray level from (x_1+1, y_1) to (x_1+1, y_1+1) is equal to the estimated gray level change per displacement along the segment s_{l+1}. That is, we have $\Delta g_{y_1} = g_{est}(x_1+1, y_1+1) - g_{est}(x_1+1, y_1) = ((g_3 + \Delta g_{s_b}) - (g_1 + \Delta g_{s_t}))/(n-1)$. Consequently, we have $\Delta g_{y_1} = g_{est}(x_1+1, y_1+1) - g_{est}(x_1+1, y_1) = ((g_3 + \Delta g_{s_b}) - (g_1 + \Delta g_{s_t}))/(n-1) = ((g_3 - g_1) + (\Delta g_{s_b} - \Delta g_{s_t}))/(n-1) = \Delta g_{s_l} + (\Delta g_{s_b} - \Delta g_{s_t})/(n-1) = \Delta g_{y_0} + (\Delta g_{s_b} - \Delta g_{s_t})/(n-1)$.

Let $\Delta g_{y_0} = \Delta g_{s_l}$ and $D_1 = (\Delta g_{s_b} - \Delta g_{s_t})/(n-1)$, the above equality can be written as

$$\Delta g_{y_1} = \Delta g_{y_0} + D_1. \tag{3}$$

Similarly, the difference of gray levels from $(x_1 + 2, y_1)$ to $(x_1 + 2, y_1 + 1)$ is represented by

$$\begin{aligned}\Delta g_{y_2} &= g_{est}(x_1 + 2, y_1 + 1) - g_{est}(x_1 + 2, y_1)\\ &= \frac{(g_3 + 2\Delta g_{s_b}) - (g_1 + 2\Delta g_{s_t})}{n-1}\\ &= \frac{(g_3 - g_1) + 2(\Delta g_{s_b} - \Delta g_{s_t})}{n-1}\\ &= \Delta g_{y_0} + 2 \times D_1.\end{aligned} \tag{4}$$

In general, from Eqs. (3) and (4), the gray level change from $(x_1 + i, y_1)$ to $(x_1 + i, y_1 + 1)$ is

$$\begin{aligned}\Delta g_{y_i} &= g_{est}(x_1 + i, y_1 + 1) - g_{est}(x_1 + i, y_1)\\ &= \frac{g_3 - g_1}{n-1} + i \times \frac{\Delta g_{S_b} - \Delta g_{S_t}}{n-1}\\ &= \Delta g_{y_0} + i \times D_1, \text{ where } 0 \leqslant i \leqslant n-1.\end{aligned} \tag{5}$$

By Eq. (5), we have the following result.

Lemma 1. *The estimated gray level at position* (x_1+i, y_1+j), $0 \leqslant i, j \leqslant n-1$, *is* $(g_1 + i \times \Delta g_{s_t}) + j \times (\Delta g_{y_0} + i \times D_1)$.

Proof. The estimated gray level at position $(x_1 + i, y_1)$ is

$$g_{est}(x_1 + i, y_1) = g_1 + i \times \Delta g_{s_t}. \tag{6}$$

By Eq. (6), the estimated gray level at position (x_1+i, y_1+j) is $g_{est}(x_1 + i, y_1 + j) = g_{est}(x_1 + i, y_1) + j \times \Delta g_{y_i} = (g_1 + i \times \Delta g_{s_t}) + j \times \Delta g_{y_i}$.

By Eq. (5), we have $g_{est}(x_1+i, y_1+j) = (g_1 + i \times \Delta g_{s_t}) + j \times \Delta g_{y_i} = (g_1 + i \times \Delta g_{s_t}) + j \times (\Delta g_{y_0} + i \times D_1)$ and completing the proof. □

By Lemma 1 and Eq. (2), we have the following result where the block in Fig. 1 is of size $n \times n$ since each block in the quadtree structure is of size $2^m \times 2^m (= n \times n)$.

Lemma 2. *The mean of Fig.* 1 *is* $\mu_{B_i} = (g_1 + g_2 + g_3 + g_4)/4$, *where* B_i *denotes the block in Fig.* 1.

Proof. The mean of the block B_i can be calculated by

$$\begin{aligned}&\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} g_{est}(x_1 + i, y_1 + j)\\ &= \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} [(g_1 + i \times \Delta g_{s_t})\\ &\quad + j \times (\Delta g_{y_0} + i \times D_1)],\\ &= n^2 \times g_1 + \frac{(n-1)n^2}{2} \times \frac{g_2 - g_1}{n-1}\\ &\quad + \frac{(n-1)n^2}{2} \times \left(\frac{g_3 - g_1}{n-1}\right) + \frac{(n-1)n}{2}\end{aligned}$$

$$\times \frac{(n-1)n}{2} \times \frac{g_4 - g_3 - g_2 + g_1}{(n-1)^2}$$

$$= \frac{n^2}{4}(g_1 + g_2 + g_3 + g_4).$$

Therefore, we have

$$\mu = \frac{\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} g_{est}(x_1 + i, y_1 + j)}{n^2}$$

$$= (g_1 + g_2 + g_3 + g_4)/4.$$

We complete the proof. □

Before deriving the closed form for calculating the variance of B_i, we need the following preliminary result.

Lemma 3. *The square sum of the estimated gray levels in the block B_i is $n^2\{(\frac{1}{2} - 2C)(g_1 g_4 + g_2 g_3) + C(g_1 + g_4)(g_2 + g_3) + [C(g_1 - g_2 - g_3 + g_4)]^2\}$ where $C = (2n-1)/6(n-1)$.*

Proof. It is known that $g_{est}(x_1 + i, y_1 + j) = (g_1 + i \times \Delta g_{s_t}) + j \times (\Delta g_{y_0} + i \times D_1)$, thus the square sum of the estimated gray levels in B_i is equal to

$$\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} [g_{est}(x_1 + i, y_1 + j)]^2$$

$$= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} [(g_1 + i\Delta g_{s_t}) + j(\Delta g_{y_0} + iD_1)]^2$$

$$= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} [(g_1 + j\Delta g_{y_0}) + (i\Delta g_{s_t} + ijD_1)]^2$$

$$= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} [(g_1 + j\Delta g_{y_0})^2 + 2(g_1 + j\Delta g_{y_0}) \times (i\Delta g_{s_t} + ijD_1) + (i\Delta g_{s_t} + ijD_1)^2]$$

$$= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \{[g_1^2 + 2jg_1\Delta g_{y_0} + (j\Delta g_{y_0})^2] + 2(ig_1\Delta g_{s_t} + ijD_1 g_1 + ij\Delta g_{y_0}\Delta g_{s_t} + ij^2\Delta g_{y_0} D_1) + [(i\Delta g_{s_t})^2 + 2i^2 j\Delta g_{s_t} D_1 + (ijD_1)^2]\}$$

$$= n^2 g_1^2 + 2n \times \frac{n(n-1)}{2} g_1 \Delta g_{y_0} + n \times \frac{n(n-1)(2n-1)}{6} \Delta g_{y_0}^2 + 2n \times \frac{n(n-1)}{2} g_1 \Delta g_{s_t} + 2 \times \frac{n(n-1)}{2} \frac{n(n-1)}{2} D_1 g_1 + 2 \times \frac{n(n-1)}{2} \frac{n(n-1)}{2} \Delta g_{y_0} \Delta g_{s_t} + 2 \times \frac{n(n-1)}{2} \frac{n(n-1)(2n-1)}{6} \Delta g_{y_0} D_1 + n \times \frac{n(n-1)(2n-1)}{6} \Delta g_{s_t}^2 + 2 \times \frac{n(n-1)}{2} \frac{n(n-1)(2n-1)}{6} \Delta g_{s_t} D_1 + \frac{n(n-1)(2n-1)}{6} \frac{n(n-1)(2n-1)}{6} D_1^2. \quad (7)$$

From $D_1 = (\Delta g_{s_b} - \Delta g_{s_t})/(n-1) = (g_4 - g_3 - g_2 + g_1)/(n-1)^2$, Eq. (7) can be written by

$$\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} [g_{est}(x_1 + i, y_1 + j)]^2$$

$$= n^2 g_1^2 + n^2 g_1(g_3 - g_1) + \frac{n^2(2n-1)}{6} \frac{(g_3 - g_1)^2}{n-1} + n^2 g_1(g_2 - g_1) + \frac{n^2}{2} g_1(g_4 - g_3 - g_2 + g_1) + \frac{n^2}{2}(g_3 - g_1)(g_2 - g_1) + \frac{n^2(2n-1)}{6(n-1)}(g_3 - g_1) \times (g_4 - g_3 - g_2 + g_1) + \frac{n^2(2n-1)}{6(n-1)}(g_2 - g_1)^2 + \frac{n^2(2n-1)}{6(n-1)}(g_2 - g_1)(g_4 - g_3 - g_2 + g_1) + \frac{n^2(2n-1)^2}{36(n-1)^2}(g_4 - g_3 - g_2 + g_1)^2$$

$$= n^2 g_1^2 + n^2 g_1 g_3 - n^2 g_1^2 + \frac{n^2(2n-1)}{6(n-1)} g_3^2 - \frac{2n^2(2n-1)}{6(n-1)} g_1 g_3 + \frac{n^2(2n-1)}{6(n-1)} g_1^2 + n^2 g_1 g_2 - n^2 g_1^2 + \frac{n^2}{2} g_1 g_4 - \frac{n^2}{2} g_1 g_3 - \frac{n^2}{2} g_1 g_2 + \frac{n^2}{2} g_1^2 + \frac{n^2}{2} g_2 g_3 - \frac{n^2}{2} g_1 g_3$$

$$
\begin{aligned}
&- \frac{n^2}{2} g_1 g_2 + \frac{n^2}{2} g_1^2 + \frac{n^2(2n-1)}{6(n-1)} g_3 g_4 \\
&- \frac{n^2(2n-1)}{6(n-1)} g_3^2 - \frac{n^2(2n-1)}{6(n-1)} g_2 g_3 \\
&+ \frac{2n^2(2n-1)}{6(n-1)} g_1 g_3 - \frac{n^2(2n-1)}{6(n-1)} g_1 g_4 \\
&+ \frac{n^2(2n-1)}{6(n-1)} g_1 g_2 - \frac{n^2(2n-1)}{6(n-1)} g_1^2 \\
&+ \frac{n^2(2n-1)}{6(n-1)} g_2^2 - \frac{2n^2(2n-1)}{6(n-1)} g_1 g_2 \\
&+ \frac{n^2(2n-1)}{6(n-1)} g_1^2 + \frac{n^2(2n-1)}{6(n-1)} g_2 g_4 \\
&- \frac{n^2(2n-1)}{6(n-1)} g_2 g_3 - \frac{n^2(2n-1)}{6(n-1)} g_2^2 \\
&+ \frac{2n^2(2n-1)}{6(n-1)} g_1 g_2 - \frac{n^2(2n-1)}{6(n-1)} g_1 g_4 \\
&+ \frac{n^2(2n-1)}{6(n-1)} g_1 g_3 - \frac{n^2(2n-1)}{6(n-1)} g_1^2 \\
&+ \frac{n^2(2n-1)^2}{36(n-1)^2} (g_4 - g_3 - g_2 + g_1)^2 \\
&= \frac{n^2}{2} g_1 g_4 + \frac{n^2}{2} g_2 g_3 + \frac{n^2(2n-1)}{6(n-1)} g_3 g_4 \\
&- \frac{2n^2(2n-1)}{6(n-1)} g_2 g_3 - \frac{2n^2(2n-1)}{6(n-1)} g_1 g_4 \\
&+ \frac{n^2(2n-1)}{6(n-1)} g_1 g_2 + \frac{n^2(2n-1)}{6(n-1)} g_2 g_4 \\
&+ \frac{n^2(2n-1)}{6(n-1)} g_1 g_3 + \frac{n^2(2n-1)^2}{36(n-1)^2} \\
&\times (g_1 - g_2 - g_3 + g_4)^2. \qquad (8)
\end{aligned}
$$

Let $C = (2n-1)/6(n-1)$. Eq. (8) can be written by

$$
\begin{aligned}
&\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} [g_{est}(x_1 + i, y_1 + j)]^2 \\
&= n^2 \{ (\tfrac{1}{2} - 2C)(g_1 g_4 + g_2 g_3) + C(g_1 + g_4)(g_2 + g_3) \\
&\quad + [C(g_1 - g_2 - g_3 + g_4)]^2 \}.
\end{aligned}
$$

We complete the proof. □

By Lemma 3, we have the following result.

Lemma 4. *The variance of B_i is $\sigma_{B_i}^2 = (\frac{1}{2} - 2C)(g_1 g_4 + g_2 g_3) + C(g_1 + g_4)(g_2 + g_3) + [C(g_1 - g_2 - g_3 + g_4)]^2 - \mu^2$.*

Proof. The variance of B_i is

$$
\begin{aligned}
\sigma_{B_i}^2 &= \frac{\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} [g_{est}(x_1 + i, y_1 + j) - \mu_{B_i}]^2}{n^2} \\
&= \frac{\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} [g_{est}(x_1 + i, y_1 + j)]^2}{n^2} - \mu_{B_i}^2.
\end{aligned}
$$

From Lemma 3, the above equality can be rewritten as

$$
\begin{aligned}
\sigma_{B_i}^2 &= (\tfrac{1}{2} - 2C)(g_1 g_4 + g_2 g_3) + C(g_1 + g_4)(g_2 + g_3) \\
&\quad + [C(g_1 - g_2 - g_3 + g_4)]^2 - \mu_{B_i}^2.
\end{aligned}
$$

We complete the proof. □

3.2. Criteria for merging two regions and the related closed forms

In a gray image, a region consists of a set of similar pixels. Initially, each leaf block corresponding to the bit '1' in the linear-tree table is considered as a new region. Before merging two promising regions, we need some criteria to determine the merging condition. The following two merging criteria are used in our proposed algorithm.

(1) *Criterion* 1: the absolute difference between the average gray values of two concerning regions must be less than the threshold.

(2) *Criterion* 2: the variance of the newly merged region must be less than the threshold.

Assume the two regions, A and B, can be merged and let region C denote the merged region of A and B. Let μ_A, μ_B, and μ_C denote the mean of region A, region B, and region C, respectively; let σ_A^2, σ_B^2, and σ_C^2 denote the variance of region A, region B, and region C, respectively. Here, we assume the size of region A is n_A; the size of region B is n_B, and the size of region C is $n_C = n_A + n_B$. We have the following two results.

Lemma 5. *The mean of the merged region C is $\mu_C = (n_A \mu_A + n_B \mu_B)/n_C$.*

Proof. By definition, the mean of C is given by

$$
\begin{aligned}
\mu_C &= \frac{\sum_{g \in C} g}{n_C} \\
&= \frac{\sum_{g_1 \in A} g_1 + \sum_{g_2 \in B} g_2}{n_C} \\
&= \frac{n_A \mu_A + n_B \mu_B}{n_C},
\end{aligned}
$$

where g denotes the gray level in the merged region C; g_1 (g_2) denotes the gray level in the region A (region B). We complete the proof. □

Lemma 6. *The variance of the merged region C is equal to* $\sigma_C^2 = (n_A\sigma_A^2 + n_B\sigma_B^2)/n_C + (n_A n_B(\mu_A - \mu_B)^2)/n_C^2$.

Proof. By Lemma 5, the variance of C is given by

$$\begin{aligned}
\sigma_C^2 &= \frac{\sum_{g\in C}(g-\mu_C)^2}{n_C}\\
&= \frac{\sum_{g_1\in A}(g_1-\mu_C)^2 + \sum_{g_2\in B}(g_2-\mu_C)^2}{n_C}\\
&= \frac{\sum_{g_1\in A}(g_1^2 - 2g_1\mu_C + \mu_C^2) + \sum_{g_2\in B}(g_2^2 - 2g_2\mu_C + \mu_C^2)}{n_C}\\
&= \frac{n_A\{(\sum_{g_1\in A} g_1^2)/n_A - 2\mu_C\mu_A + \mu_C^2\} + n_B\{(\sum_{g_2\in B} g_2^2)/n_B - 2\mu_C\mu_B + \mu_C^2\}}{n_C}\\
&= \frac{n_A\{\sigma_A^2 + (\mu_A-\mu_C)^2\} + n_B\{\sigma_B^2 + (\mu_B-\mu_C)^2\}}{n_C}\\
&= \frac{n_A\sigma_A^2 + n_B\sigma_B^2}{n_C} + \frac{n_A(n_B(\mu_A-\mu_B)/n_C)^2 + n_B(n_A(\mu_A-\mu_B)/n_C)^2}{n_C}\\
&= \frac{n_A\sigma_A^2 + n_B\sigma_B^2}{n_C} + \frac{(n_An_B^2 + n_Bn_A^2)((\mu_A-\mu_B)/n_C)^2}{n_C}\\
&= \frac{n_A\sigma_A^2 + n_B\sigma_B^2}{n_C} + \frac{n_An_B(\mu_A-\mu_B)^2}{n_C^2},
\end{aligned}$$

where g denotes the gray level in the merged region C; g_1 (g_2) denotes the gray level in region A (region B). We complete the proof. □

From Lemmas 2, 4, 5, and 6, we have the first main result.

Theorem 1. *The mean/variance of any block can be calculated in $O(1)$ time. The mean/variance of any newly merged block also can be calculated in $O(1)$ time.*

From the viewpoint of practical implementation, instead of applying Theorem 1, we calculate the related mean and variance by the direct way when the block is of size 2×2.

If we use the sample variance, it yields

$$\begin{aligned}
\sigma_C^2 &= \frac{\sum_{g\in C}(g-\mu_C)^2}{n_C-1}\\
&= \frac{\sum_{g\in C}(g^2 - 2g\mu_C + \mu_C^2)}{n_C-1}\\
&= \frac{\sum_{g\in C} g^2 - 2\mu_C\sum_{g\in C} g + n_C\mu_C^2}{n_C-1}\\
&= \frac{\sum_{g\in C} g^2 - n_C\mu_C^2}{n_C-1}\\
&= \frac{\sum_{g_1\in A} g_1^2 + \sum_{g_2\in B} g_2^2 - (1/n_C)(\sum_{g_1\in A} g_1 + \sum_{g_2\in B} g_2)^2}{n_C-1}.
\end{aligned}$$

Since by Lemma 2, the sum of the data values in and the sum of the squared data values in the two constituent regions can be calculated in $O(1)$ time, Theorem 1 is still true when using the sample variance.

3.3. The required data structure

According to the S-tree representation described in Section 2 and based on the data structure used in Ref. [6], this subsection presents the related data structure which will be used in our proposed region-segmentation algorithm on the S-tree representation. Since our region-segmentation algorithm extends the results in Ref. [6] from the map image to the gray image, the data structure used in our algorithm is somewhat different from those in Ref. [6].

For exposition, as shown in Fig. 3, let us first see the final segmented regions of Fig. 2(a). In Fig. 3, there are six segmented regions, namely A, B, C, D, E, and F. For each region, we use a sequence of corners' coordinates in a clockwise manner to record the boundary of the region. For example, the boundary of region A in Fig. 3 is recorded by

$$\{(0,0),(12,0),(12,4),(8,4),(8,6),(6,6),(6,8),(4,8),(4,10),(2,10),(2,6),(0,6)\}.$$

Following the similar notations used in Ref. [6], in what follows, three definitions are given.

Definition 1. A region is active if it will be merged with the other region. Otherwise, a region is inactive.

Definition 2. An active region or inactive region is represented by the boundary of that region and the boundary

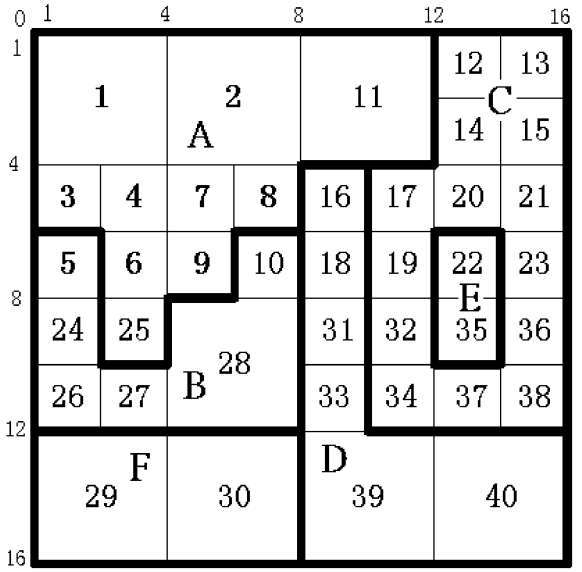

Fig. 3. An example for decomposing the image into some regions.

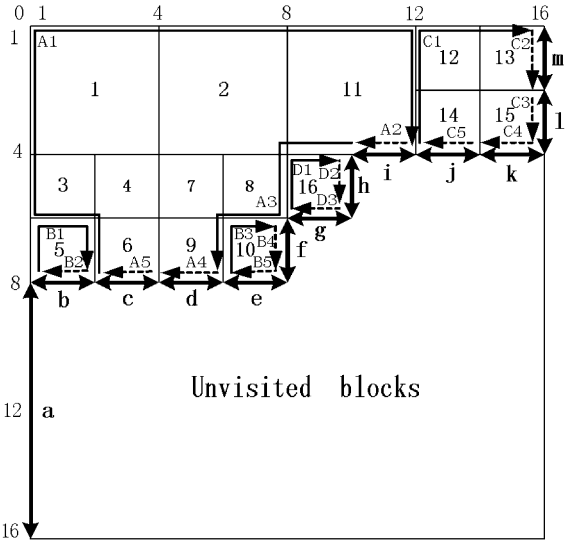

Fig. 4. An example of waveform, inactive edges, active edges.

is composed of consecutive edges, each edge being active or inactive. For an active edge, it will be merged with the neighboring inactive edge or become a new inactive edge if the two concerning active regions connecting the active edge as the common edge cannot be merged.

Consider Fig. 3 again. Initially, we have 40 active regions, but eventually there are only six inactive regions. As mentioned in Section 2, we represent the compressed gray image using the S-tree representation consisting of linear-tree table and the color table. We visit the linear-tree table bit-by-bit from left to right. Once the bit '1' is visited, it implies that we visit a leaf node, i.e., block in the quadtree, and then we access its corresponding four corners' gray values in the color table. Initially, each visited bit '1' can be viewed as a new region, i.e., active region, which may be merged with the other active region later.

Definition 3. The waveform consisting of some segments is the border between those visited quadtree blocks corresponding those 1s in the S-tree representation and those unvisited blocks.

Let us return to Fig. 2(a). Before performing the region-segmentation, the waveform is composed of two segments, one segment from position $(0,0)$ to position $(0,16)$ and the other segment from $(0,0)$ to $(16,0)$ since those forty initial blocks indexed by $1,2,\ldots,$ and 40 are not visited. We now define the data structure used for the waveform. We use a double link list for representing the waveform. Each node in the double link list denotes one segment and the **Segment** record contains four fields: $\{PreLink, Length, ActiveELink, SucLink\}$ where the fields *SucLink* and *PreLink* are pointers denoting the next segment and the previous segment in the double link list; the *Length* field denotes the segment's length, and the ActiveELink field points to the **Edge** record which contains five fields: $\{PreLink, First, Last, Reg, SucLink\}$.

We now take an example to explain how the data structure defined for segments are used. Under the S-tree representation, as shown in Fig. 4, suppose we have scanned the blocks $1,2,3,\ldots,$ and 16, then the waveform is like a wave consisting of thirteen segments where each segment is denoted by a solid line bounded by two arrows. Thus, the data structure for the waveform in Fig. 4 can be depicted in Fig. 5. In Fig. 5, the data structure of each segment in the waveform is the record with four fields. The second node *b* denotes the second segment starting from position $(0,8)$ to $(2,8)$ and its related record is depicted in node *b* where the *PreLink* field points to node *a*; the *Length* field contains the second segment's length, i.e., 2; the *ActiveELink* field points to the active edge *B*2.

Considering Fig. 4 again, we further explain how the data structure defined for edges are used. Let us consider the active region (see Definition 2) containing the blocks with indices 1, 2, 3, 4, 6, 7, 8, 9, and 11. The data structure for inactive edge and active edge can be depicted in Fig. 6. In Fig. 6, the data structure of each edge in the active region is the record with five fields. The second node denotes the second edge starting from position $(12,4)$ to $(10,4)$ where the *PreLink* field points to the inactive edge *A*1; both the *First* and *Last* fields contain NULL's; the *Reg* field points to region 11, i.e., block 11; the *SucLink* field points to the inactive edge *A*3. Looking at the node named 'inactive edge *A*1, the *First* and *Last* fields are used to point the starting position $(2,8)$ and the ending position $(12,4)$. Here

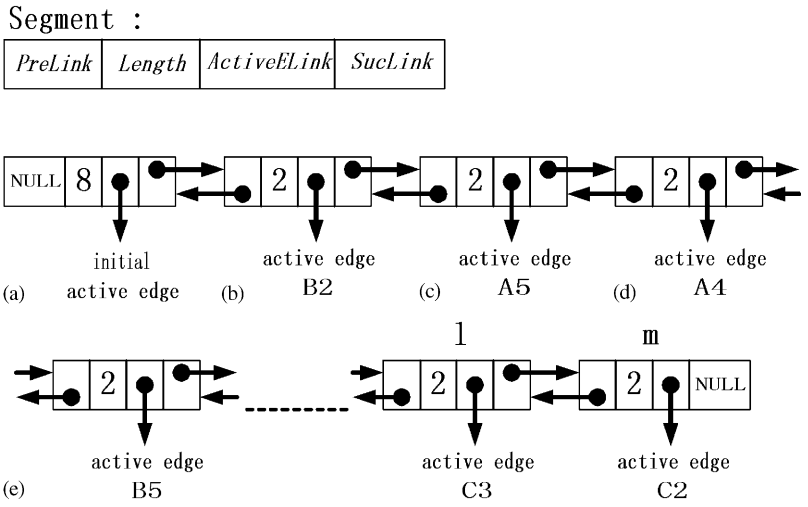

Fig. 5. The data structure for the waveform in Fig. 4.

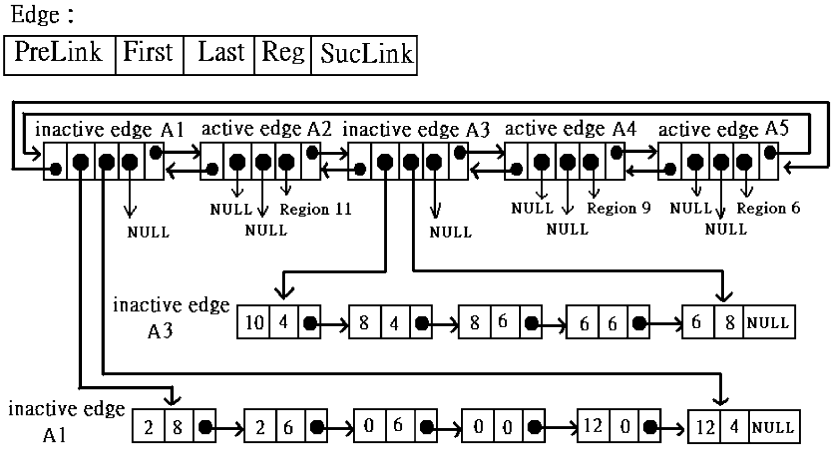

Fig. 6. The data structure for inactive edge and active edge.

positions $(2,8)$, $(2,6)$, $(0,6)$, $(0,0)$, $(12,0)$, and $(12,4)$ are the corner-vertices of inactive edge $A1$. Each **Corner-Vertex** record contains three fields: $\{X, Y, Next\}$ where X and Y denote the coordinates; the symbol *Next* points to the next corner-vertex.

According to the DFS scanning order employed in the *linear tree table*, let us look at Fig. 4 again. After processing block 16, there are five active regions. For each active region, there exists at least one active edge commonly shared by one segment on the waveform. In Fig. 4, one active region contains the quadtree blocks indexed by 1, 2, 3, 4, 6, 7, 8, 9, and 11. Another active region contains the quadtree blocks indexed by 12, 13, 14, and 15. The remaining three active regions consist of the quadtree blocks 5, 10, and 16, respectively.

If all the edges of one active region become inactive, the status of that region becomes inactive and will never be changed. For any two active regions, if we want to merge them, we apply the union-find algorithm [15,16]. Now we define the **Region** record to specify one region. One region record contains seven fields: $\{Mean, Var, Size, Father, Count, SegmentCount, EdgeLink\}$ where the symbols *Mean*, *Var*, and *Size* denote the mean, the variance, and the side-length of the block, i.e., the number of the pixels in the region. These three fields are used to support for merging two active regions.

The field *Father* is an address to point to the father of this region. The field *Count* is used to count the number of region-nodes which are descendants of this region. The above fields are used to support the union-find algorithm.

The field *SegmentCount* denotes the number of edges commonly shared by the segments of the waveform. The field *EdgeLink* points to the edge of this region, which can be used to trace the boundary information of this region.

3.4. The proposed algorithm

After describing the necessary data structure, based on the related closed forms for merging two regions (see Sections 3.1 and 3.2), our proposed region-segmentation algorithm on the S-tree representation is described in this subsection.

Since the input gray image has been compressed into the S-tree representation which consists of two arrays, namely the *liner-tree table* and the color table, we scan the linear-tree table sequentially. Upon scanning the bit '0', four recursive procedures are called and the four recursive procedures are corresponding to *nw*, *ne*, *sw*, and *se* quadrants, respectively. Upon scanning the bit '1', a new region is created and then we perform the merging work for the created region and its neighboring active regions. If that created region can be merged with some active regions, we further update the edge information and the statistical measures in the newly merged region.

In order to explain our proposed algorithm, we follow the similar recursive approach in Ref. [6] to describe our proposed region-segmentation algorithm. Since the concerning blocks and the merging criterion are different, the parameters used in some procedures are some different from those in Ref. [6]. The main program of our proposed algorithm is a recursive procedure named **Region_Segm** where the related parameters will be described after the following procedure:

```
/* Input: UpperLeft, X_left, Y_upper, Size */
/* Output: UpperRight, PreLowerLeft */
/* SegmentPTR is the pointer type of Segment */
Region_Segm(UpperLeft, UpperRight, PreLowerLeft,
  X_left, Y_upper, Size)
  value SegmentPTR UpperLeft;
  reference SegmentPTR UpperRight, PreLowerLeft;
  value integer X_left, Y_upper, Size;
  SegmentPTR UR, PLL, DUMMY; /* local variable */
  boolean bit; /* bit is '1'(leaf) or '0'(nonleaf) */
begin
  if Length(UpperLeft) > Size then SPLIT(UpperLeft,
    Size);
  if Length(SucLink(UpperLeft)) > Size then SPLIT
    (SucLink(UpperLeft), Size);
  bit = Get_bit( );
  if bit = ='0' then
  begin
  Region_Segm(UpperLeft, UR, PLL, X_left,
  Y_upper, Size/2); /* nw */
  Region_Segm(UR, UpperRight, DUMMY, X_left
    +Size/2, Y_upper, Size/2); /* ne */
  Region_Segm(PLL, UR, PreLowerLeft, X_left,
  Y_upper + Size/2, Size/2); /* sw */
  Region_Segm(UR, DUMMY, DUMMY, X_left + Size/2,
    Y_upper + Size/2, Size/2); /* se */
  end
else /* leaf node */
  begin
  FourGray Color; /* record the four corners' gray
    levels of leaf node */
  Color = Get_color( );
  Leaf_Operation(UpperLeft, UpperRight,
  PreLowerLeft,
  X_left, Y_upper, Size, Color);
  end
end
```

In the above recursive program, there are six parameters which are used to maintain the segments on the waveform; calculate the size and location for each block, and update the set of regions. Whenever the bit '1' in the linear-tree table is scanned, we want to find the current waveform's segments which are shared commonly to the adjacent block in the west and north directions. When Region_Segm is called, *UpperLeft* points to the uppermost segment along the left segments of the current quadrant before processing this quadrant. When Region_Segm calls Leaf_Operation, it passes this pointer to Leaf_Operation. Whenever Region_Segm and Leaf_Operation are called, they return two pointers, *UpperRight* and *PreLowerLeft*. *UpperRight* is a pointer to point to the uppermost segment along the right segments of the current quadrant after this quadrant has been processed and the waveform has been updated. For any leftmost block in the quadrant, its *PreLowerLeft* is a pointer to point to the previous segment on the waveform. Now we take a simple example to demonstrate the above related parameters although it is applicable to more complex case.

For example, when the procedure Leaf_Operation is called to process block 28 in Fig. 2(a). *UpperLeft* points to the segment between block 28 and block 25. After finishing the procedure Leaf_Operation, *UpperRight* (*PreLowerLeft*) points to the segment between block 28 and block 31, 33 (block 27 and block 29). Considering another example, if the bit '0' (corresponding to the quadrant P_0) is got and suppose the quadrant P_0 has four sons, i.e., blocks 31, 32, 33, and 34. After Region_Segm has been processed once for the scanned bit '0', *PreLowerLeft* points to the segment between block 30 and block 39 and *UpperRight* points to the segment between block 32 and block 35. For the quadrant P_0, the four pointers with respect to the four directions are determined by the following four assignments:

1. $UpperLeft(nw(P_0)) = UpperLeft(P_0)$.
2. $UpperLeft(ne(P_0)) = UpperRight(nw(P_0))$.
3. $UpperLeft(sw(P_0)) = PreLowerLeft(nw(P_0))$.
4. $UpperLeft(se(P_0)) = UpperRight(sw(P_0))$.

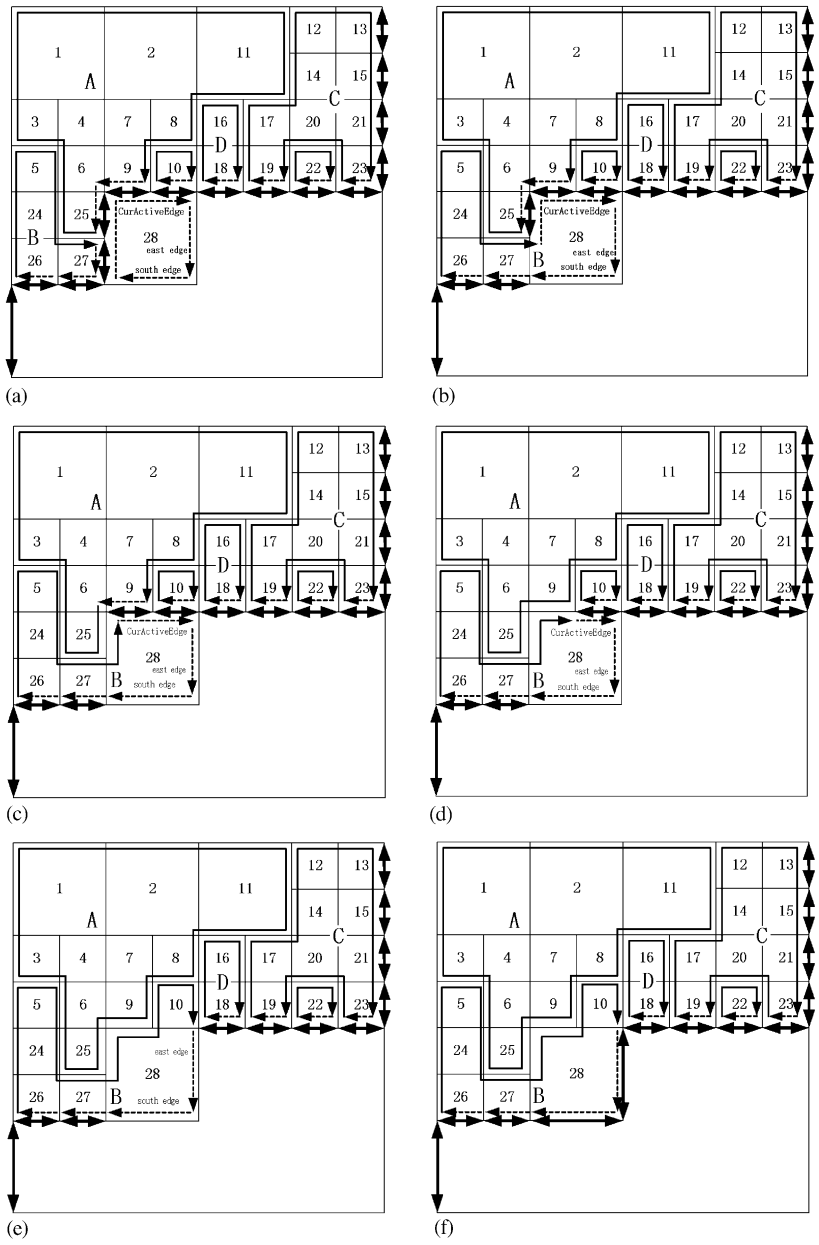

Fig. 7. A simulation of Leaf_Operation for block 28: (a) create block 28; (b) after merging block 28 and active region *B*; (c) after merging active region *A* and *B*; (d) after merging active region *A* and *B*; (e) after merging active region *B* and block 10; (f) update segments on the waveform.

Region_Segm will check whether the segment on the waveform is longer than the corresponding left edge or top edge of the current quadrant. This situation can arise if the block is smaller than its western or northern neighboring block. For example, in Fig. 2(a), when block 3 is processed, its top edge is shorter than the segment which is the line segment from $(0,4)$ to $(4,4)$. In this case, the left and/or top segment is split by half in order to perform the merging work.

The function call, Get_bit( ), in the procedure Region_Segm is used to get the next bit from the linear-tree table. If the bit '1' is got, we perform the function call Get_color( ) to record the four corners' gray values of the corresponding block in the color table and call the procedure Leaf_Operation. By Theorem 1, we can calculate the mean/variance using the four corners' gray values and the size of the current block in $O(1)$ time. If we get bit

Table 1
Execution time performance comparison

Image	Camera	Boat	Lena	F16
NP	262144	262144	262144	262144
NB ($\varepsilon = 10$)	13903	32776	24955	26020
NB ($\varepsilon = 15$)	10309	26467	17689	20677
NMB ($\varepsilon = 10$)	8912	28908	19052	21692
NMB ($\varepsilon = 15$)	5188	21696	12040	16316
linear_tree_table ($\varepsilon = 10$)	18537 (bits)	43701 (bits)	33273 (bits)	34693 (bits)
linear_tree_table ($\varepsilon = 15$)	13745 (bits)	35289 (bits)	23585 (bits)	27569 (bits)
color_table ($\varepsilon = 10$)	444896 (bits)	1048832 (bits)	798560 (bits)	832640 (bits)
color_table ($\varepsilon = 15$)	329888 (bits)	846944 (bits)	566048 (bits)	661664 (bits)
bpp ($\varepsilon = 10$)	1.7678 (bit)	4.1676 (bit)	3.1731 (bit)	3.3086 (bit)
bpp ($\varepsilon = 15$)	1.3108 (bit)	3.3654 (bit)	2.2492 (bit)	2.6292 (bit)
PSNR ($\varepsilon = 10$)	40.4728	40.8214	39.0718	39.8347
PSNR ($\varepsilon = 15$)	37.9068	37.1772	36.1628	36.9471
T_{ours} ($\varepsilon = 10$)	0.10	0.23	0.18	0.19
T_{ours} ($\varepsilon = 15$)	0.08	0.19	0.14	0.16
T_{LBL} [8]	0.33	0.34	0.33	0.33
T_{DAC} [8]	0.39	0.39	0.39	0.39
$\frac{T_{LBL}-T_{ours}\ (\varepsilon=10)}{T_{LBL}}$	69%	32%	45%	42%
$\frac{T_{DAC}-T_{ours}\ (\varepsilon=10)}{T_{DAC}}$	74%	41%	53%	51%
$\frac{T_{LBL}-T_{ours}\ (\varepsilon=15)}{T_{LBL}}$	75%	44%	57%	51%
$\frac{T_{DAC}-T_{ours}\ (\varepsilon=15)}{T_{DAC}}$	79%	51%	64%	58%

'0' from the linear-tree table, then Region_Segm calls itself recursively to process the four corresponding *nw*, *ne*, *sw*, and *se* quadrants. In the procedure Region_Segm, the variables (X_{left}, Y_{upper}) are used to denote coordinates of the upper-left corner of the quadrant and the variable *Size* represents the size of the quadrant.

When the procedure Leaf_Operation read the input value *UpperLeft*, we first walk down along the left side of the current block until we find the corresponding segment pointed by *LowerLeft*. For example, in Fig. 7(a), *LowerLeft* points to the segment between the block 28 and the active region *B*. Note that when reading the bit '1' from the S-tree representation, a new record with type Region is created for the created block and the Region record consists of three active edges, namely the east edge of this block (see Fig. 7(a)), the south edge of this block (see Fig. 7(a)), and the northwest active edge (see the CurActiveEdge in Fig. 7(a)).

By using the pointer SucLink, Leaf_Operation processes the segments, which are adjacent to the left side and the top side of the current block, from the left-lowest one to the upper-rightmost one (see Figs. 7(b)–(e)). Upon processing one segment, according to the merging criterion described in Section 3.2, we can determine whether the adjacent active region should be merged with the current region or not. If the adjacent active region can be merged with the current region, a union-find operation [15,16] is applied. In each union-find operation, both weight-balancing strategy and path-compression strategy are used. After that, by Lemmas 5 and 6, the statistical measures of the newly merged region are calculated in $O(1)$ time. Simultaneously, the edges information of the merged region is saved in the root of the Region record who is a tree structure. Let r denote the newly merged region record. When SegmentCount(r)=0 and Count(r)=0, the current active region does become the inactive region and can be output. In fact, we output the corner-vertices of this inactive region. When the current block has finished the merging work, we update the waveform (see Fig. 7(f)). Eventually, we have the following main result.

Theorem 2. *On the compressed gray image using the S-tree representation, our proposed region-segmentation can be performed in $O(B\alpha(B))$ time where B denotes the number of* 1*s in the linear-tree table of the S-tree representation. In fact, B is also the number of blocks.*

Proof. In our proposed region-segmentation algorithm, once the bit '0' is read from the linear-tree table of the S-tree representation, we call the procedure Region_Segm, so totally it takes at most $O(A)$ times to call the procedure Region_Segm where A denotes the number of 0s in the linear-tree table. Once the bit '1' is read from the linear-tree table of the S-tree representation, we call the procedure Leaf_Operation, so totally it takes at most $O(B)$ times to call the procedure Leaf_Operation where B denotes the

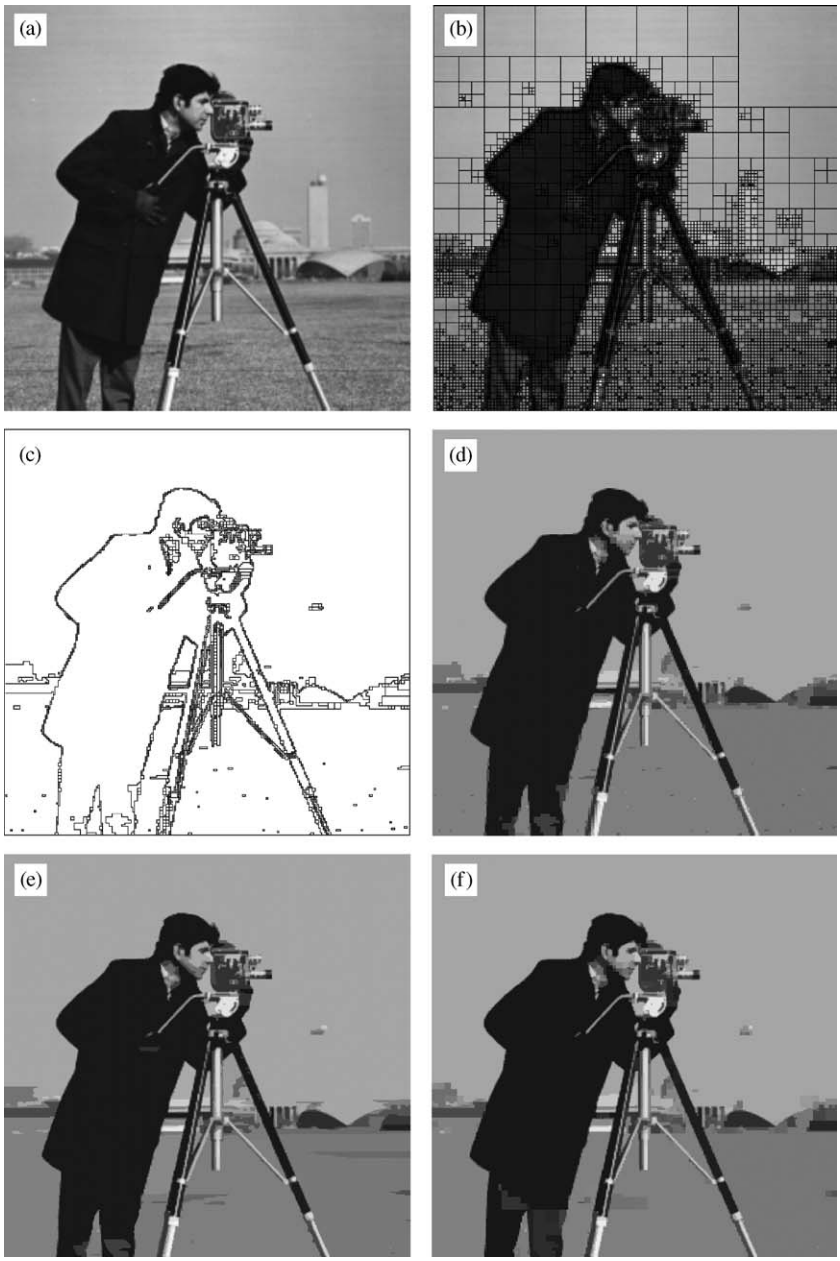

Fig. 8. The segmentation comparison for 'camera' image: (a) the original 'camera' image; (b) the partitioned blocks for $\varepsilon = 10$; (c) the segmented regions using our method (for each region, $\mu < 30$ and $\sigma^2 < 225$); (d) each region is displayed by the mean value of that region using our method; (e) the segmentation result by using the line by line strategy [8] (for each region, $\mu < 30$ and $\sigma^2 < 225$); (f) the segmentation result by using the divide and conquer strategy [8] (for each region, $\mu < 30$ and $\sigma^2 < 225$).

number of 1s in the linear-tree table. When reading '1' from the linear-tree table, a new block is created. It can be easily checked that there are at most four segments on the waveform which are adjacent to the edges of the newly created block. Therefore, there are at most $4B$ times for applying the merging operation to merge the current block and the neighboring active regions via the segments' guidance. Since each merging work only needs $O(1)$ time by Theorem 1, employing the union-find operations successively, it implies that the total time complexity is $O(B\alpha(B)+A)$. Since $A < B$, the total time complexity can be expressed as $O(B\alpha(B))$. We complete the proof. □

4. Experimental results

In our experiments, four real images, namely the camera, the boat, the Lena, and the F16, are used to test the performance of our proposed region-segmentation algorithm on the S-tree representation and the two variants of the previous fastest region-segmentation algorithm by Fiorio and Gustedt [8]. Each image is of size 512×512 and it requires 262144 bits. All experiments are performed on the IBM Pentium III microprocessor with 667 MHz and 128 MB RAM. The operation system is MS-Windows 2000 and the program developing environment is Borland C++ Builder 5.0.

For the four tested images, the first experiment is used to evaluate the execution time performance among the concerning algorithms. Experimental results are shown in Table 1 where the symbol NP denotes the number of pixels required in the image; the symbol NB denotes the number of blocks, i.e., the number of 1s, in the S-tree representation; the symbol NMB denotes the number of 2×2 blocks in the S-tree representation; the symbol T_{ours} ($\varepsilon = 10$) denotes the execution time in terms of seconds required in our proposed algorithm for $\varepsilon = 10$; T_{LBL} denotes the execution time required in the line-by-line version of the algorithm in Ref. [8]; T_{DAC} denotes the execution time required in the divide-and-conquer version of the algorithm in Ref. [8]. The execution time improvement ratio of our proposed algorithm over the previous line-by-line-based (divide-and-conquer-based) algorithm is measured by $(T_{LBL} - T_{ours})/T_{LBL}$ ($(T_{DAC} - T_{ours})/T_{DAC}$). The image quality is denoted by PSNR [17] which is defined by

$$\mathrm{PSNR} = 10 \log_{10} \left(\frac{255^2}{\mathrm{MSE}} \right),$$

where MSE is the mean square error between the decompressed image and the original image. The number of bits required in one pixel is denoted by bpp (bits per pixel). From Table 1, it is observed that for $\varepsilon = 10$, the execution time improvement ratio of our proposed algorithm over the LBL (DAC) is 47% (55%); for $\varepsilon = 15$, the execution time improvement ratio of our proposed algorithm over the LBL (DAC) is 57% (63%). From the rows for T_{LBL} and T_{DAC}, the required execution time is linear and rather stable in terms of NP. However, the execution time required in our proposed algorithm is dependent on the number of blocks, i.e., NB, and is nearly linear and rather stable since $\alpha(B)$ is a very slowly growing function.

To save space, we only show the resulting segmented regions for the 'camera' image. Fig. 8(a) shows the original test image 'camera'. Fig. 8(b) is the partitioned blocks for $\varepsilon = 10$ via the QS-based method. Fig. 8(c) shows the inactive edges for each segmented region. For displaying the segmentation effect of our proposed algorithm, we use the mean gray value of each region to demonstrate that region (see Fig. 8(d)). In Figs. 8(e) and (f), we show the two segmentation results using the two methods [8]. From the human visual point, the segmentation result of our proposed method is quite competitive to the two variants in Ref. [8]. However, our proposed method is much faster than the two variants [8].

5. Conclusions

We have presented the efficient region-segmentation algorithm on the compressed gray image using the S-tree representation directly. The compressed gray image is presented by the quadtree and shading format. We first derive some closed forms for computing the mean/variance of any one block and for calculating the new mean/variance of the newly merged region. Then, a nearly linear-time region-segmentation algorithm is presented, i.e., the time complexity of our proposed algorithm depends on the number of 1s in the S-tree representation, say B, and the required time complexity is $O(B\alpha(B))$ where $\alpha(B)$ is the inverse of the Ackerman's function and nearly can be viewed as a constant. With the same time complexity, our results extend the previous pioneering results by Dillencourt and Samet [6] from the map domain to the gray domain. In addition, under four real images, experimental results show that our proposed algorithm has about 55.4% execution time improvement ratio when compared to the previous fastest two variants of the algorithm by Fiorio and Gustedt [8] whose $O(N^2)$-time algorithm is run on the original $N \times N$ gray image. It is an interesting research problem to solve the region-segmentation on the S-tree representation using $O(B)$ time.

References

[1] R.C. Gonzalez, R.E. Woods, Digital Image Processing, Chapter 10: Image Segmentation, 2nd Edition, Prentice-Hall, New York, 2002.

[2] R.M. Haralick, L.G. Shapiro, Computer and Robot Vision, Addison-Wesley, New York, 1992.

[3] J. Llados, E. Marti, J.J. Villanueva, Symbol recognition by error-tolerant subgraph matching between region adjacency graphs, IEEE Trans. Pattern Anal. Mach. Intell. 23 (10) (2001) 1137–1143.

[4] R.M. Haralick, L.G. Shapiro, Survey: image segmentation techniques, Comput. Vision Graphics Image Process. 29 (1985) 100–132.

[5] P.K. Sahoo, S. Soltani, A.K.C. Wong, A survey of thresholding techniques, Comput. Vision Graphics Image Process. 41 (1988) 233–260.

[6] M.B. Dillencourt, H. Samet, Using topological sweep to extract the boundaries of regions in maps represented by region quadtrees, Algorithmica 15 (1) (1996) 82–102.

[7] H. Samet, The Design and Analysis of Spatial Data Structures, Addison-Wesley, Reading, MA, 1990.

[8] C. Fiorio, J. Gustedt, Two linear time union-find strategies for image processing, Theoret. Comput. Sci. 154 (2) (1996) 165–181.

[9] J. Gustedt, Efficient union-find for planar graphs and other sparse graph classes, Theoret. Comput. Sci. 203 (1998) 123–141.

[10] R. Distasi, M. Nappi, S. Vitulano, Image compression by B-tree triangular coding, IEEE Trans. Commun. 45 (9) (1997) 1095–1100.
[11] W.B. Pennebaker, J.L. Mitchell, JPEG: Still Image Data Compression Standard, New York, 1993.
[12] W.D. Jonge, P. Scheuermann, A. Schijf, S^+-trees: an efficient structure for the representation of large pictures, Comput. Vision Image Understanding 59 (1994) 265–280.
[13] J.D. Foley, A.V. Dam, S.K. Feiner, J.F. Hughes, Computer Graphics, Principle, and Practice, 2nd Edition, Addison-Wesley, Reading, MA, 1990.
[14] K.L. Chung, J.G. Wu, Improved image compression using S-tree and shading approach, IEEE Trans. Commun. 48 (5) (2000) 748–751.
[15] R.E. Tarjan, Efficiency of a good but not linear set union algorithm, J. ACM 22 (2) (1975) 215–225.
[16] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, 2nd Edition, McGraw-Hill, New York, 2001.
[17] K. Sayood, Introduction to Data Compression, 2nd Edition, Morgan Kaufmann, New York, 2000.

About the Author—KUO-LIANG CHUNG received the B.S., M.S., and Ph.D. degrees in Computer Science and Information Engineering from National Taiwan University in 1982, 1984, and 1990, respectively. From 1984 to 1986, he was a soldier. From 1986 to 1987, he was a research assistant in the Institute of Information Science, Academic Sinica. He has been a Professor in the Department of Computer Science and Information Engineering at National Taiwan University of Science and Technology since 1995. He has been the Chairman since 2003. Prof. Chung received the Distinguished Professor Award from the Chinese Institute of Engineers in May 2001. He has been the IEEE senior member since 2001. Prof. Chung received the Outstanding I.T. Elite Award from the R.O.C. Information Month in November 2003. His research interests include image compression, image processing, video processing, video compression, coding theory, and algorithms.

About the Author—HSU-LIEN HUANG received the M.S. degree in Information Management from National Taiwan University of Science and Technology in 1999 and 2001, respectively. Her research interests include image compression, image processing, and algorithms.

About the Author—HSUEH-I LU received his B.S. and M.S. degrees in Computer Science and Information Engineering from National Taiwan University, Taipei, Taiwan, in 1986 and 1990, respectively. He received his M.S. and Ph.D. degrees in Computer Science from Brown University, Providence, RI, USA in 1992 and 1997, respectively. In 1997, he joined Department of Computer Science and Information Engineering, National Chung-Cheng University, Chia-Yi, Taiwan. In 1999, he joined Institute of Information Science, Academia Sinica, Taipei, Taiwan as an assistant research fellow. He was promoted to an associate research fellow in 2003. Dr. Lu's research focuses on design and analysis of algorithms. In 2002, he was awarded Ta-You Wu Memorial Award from National Science Council, Taiwan.