Real-Time Imaging 8, 137–144 (2002)
doi:10.1006/rtim.2001.0266, available online at http://www.idealibrary.com on IDEAL®

# Fast Computation of Moments on Compressed Grey Images using Block Representation

In image processing, moments are useful tools for analyzing shapes. Suppose that the input grey image with size $N \times N$ has been compressed into the compressed image using the block representation, where the number of blocks used is $K$, commonly $K < N^2$ due to the compression effect. This paper presents an efficient $O(N\sqrt{K})$-time algorithm for computing moments on the compressed image directly. Experimental results reveal a significant computational advantage of the proposed algorithm, while preserving a high accuracy of moments and a good compression ratio. The results of this paper extend the previous results in [7] from the binary image domain to the grey image domain.



**Kuo-Liang Chung,*,[1] Wen-Ming Yan[2] and Zhi-Hor Liao[1]**

[1]*Department Computer Science and Information Engineering, National Taiwan University of Science and Technology, No. 43, Section 4, Keelung Road, Taipei, Taiwan 10672, R. O. C. E-mail: klchung@cs.ntust.edu.tw*

[2]*Department of Computer Science and Information Engineering, National Taiwan University, No. 1, Section 4, Roosevelt Road, Taipei, Taiwan 100, R. O. C. E-mail: ganboon@csie.ntu.edu.tw*

## Introduction

In image analysis, moments are useful tools for analyzing shapes. For an $N \times N$ image, let $g(x, y)$ denote the grey level of the pixel at location $(x, y)$ for $0 \leq x, y \leq N-1$. The $(p+q)$-order moment is defined as

$$m_{pq} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} x^p y^q g(x, y) \tag{1}$$

In fact, zero- to third-order moments are most widely used in applications [1–6]. Recently, Spiliotis and Mertzios [7] presented a very efficient algorithm for computing moments on binary images using the block representation. Their algorithm first partitions the given binary image into a set of rectangular blocks. Then some analytic formulas are derived to speed up the computation of moments. Based on the compressed grey image using the STC method, which is described in the next paragraph, the motivation of this research is to extend the results [7] from the binary image domain to the grey image domain. For convenience, the concerning low-order moments are called the moments and the compressed grey image using the STC method is called the compressed image throughout this paper in order to avoid confusion.

Previously, based on the B-tree triangular approach, Distasi *et al.* [8] presented an efficient image compression method. Their method has a shorter execution time than that of the standard JPEG [9], although the bit rate is higher by a factor of about 2. Based on the S-tree data

*Author to whom all correspondence should be addressed.

1077-2014/02/$35.00 

structure [10] and the Gouraud shading technique [11], an improved compression method called the STC method [12] is presented to partition the given image into a set of blocks. Without the mosaic effect, the STC method has shorter encoding/decoding time than [8], 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 [13,14] from the binary image domain to the grey image domain.

Suppose that the given grey image has been compressed into the compressed image with $K$ blocks, $K < N^2$. This paper presents an efficient $O(N\sqrt{K})$-time algorithm for computing moments on the compressed image directly. A detailed time complexity analysis is also provided. Some real experiments are carried out to demonstrate the significant computational advantage of the proposed algorithm while preserving a high accuracy of moments and good compression ratio. The results of this paper can be viewed as the extension of [7] from the binary image domain to the grey image domain.

## Compressed Images

In the STC method, the original grey image is partitioned into several homogeneous blocks based on the bintree decomposition principle, then the S-tree representation is used to represent these homogeneous blocks. A quantified definition of a homogeneous block will be defined in Eqn (2).

The S-tree representation is based on the breadth-first search (BFS) technique [15] and consists of two tables, namely the linear-tree table and the colour table. Following the BFS tree traversal method, the bintree representation is based on dividing the image into two equal-sized subimages recursively. At each division step, the partition is alternated between the $x$- and $y$-axis. If the subimage is not a homogeneous block, then it is subdivided into two equal-sized subimages until all the homogeneous blocks are obtained in the S-tree representation, traversing the bintree in the 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 grey-levels of the four corner pixels of the related homogeneous block, say ($g_1$, $g_2$, $g_3$,

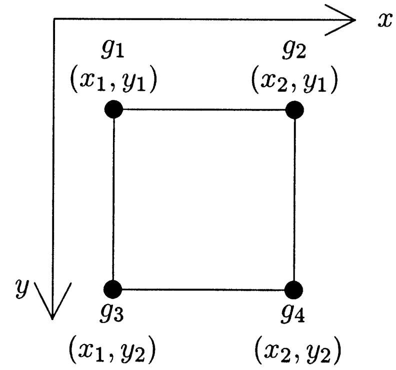



**Figure 1.** Homogeneous block.

$g_4$). The sequence of these ordered values is stored in the colour table.

We now give a quantified definition for the homogeneous block as shown in Figure 1. Using the Gouraud shading method, the estimated grey-level of the pixel at $(x, y)$, $g_{est}(x, y)$, in the homogeneous block is calculated by

$$g_{est}(x, y) = g_5 + \frac{g_6 - g_5}{y_2 - y_1}(y - y_1) \tag{2}$$

where

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

and

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

Given a specified error tolerance $\varepsilon$, if the following image quality condition holds

$$|g(x, y) - g_{est}(x, y)| \leq \varepsilon$$

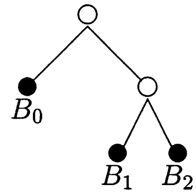
then it holds for all the estimated pixels at positions $(x, y)$'s in the block for $x_1 \leq x \leq x_2$ and $y_1 \leq y \leq y_2$, where $g(x, y)$ denotes the real grey-level of the pixel at $(x, y)$, then the block is called a homogeneous block.

Given an $8 \times 8$ grey image as shown in Figure 2(a), suppose that the error tolerance is set to $\varepsilon = 5$ and the four corners of Figure 2(a) have the four grey-levels $g_1(0, 0) = 4$, $g_2(7, 0) = 4$, $g_3(0, 7) = 25$, and $g_4(7, 7) = 4$. By Eqn (2), the estimated grey-level of the pixel at $(1, 0)$ is calculated by

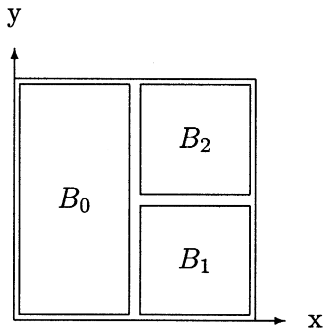
$$g_{est}(1, 0) = 4 + \frac{22 - 4}{7 - 0}(0 - 0) = 4$$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 4 | 14 | 24 | 34 | 1 | 2 | 4 | 4 |
| 8 | 18 | 32 | 42 | 4 | 6 | 8 | 10 |
| 10 | 24 | 38 | 52 | 8 | 10 | 14 | 16 |
| 14 | 28 | 46 | 60 | 10 | 14 | 18 | 22 |
| 16 | 34 | 52 | 70 | 1 | 8 | 16 | 22 |
| 20 | 38 | 60 | 78 | 4 | 8 | 12 | 16 |
| 22 | 44 | 66 | 88 | 8 | 8 | 10 | 10 |
| 25 | 48 | 74 | 97 | 10 | 8 | 6 | 4 |

(a)




(c)




(b)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 4 | 14 | 24 | 34 | 1 | 2 | 3 | 4 |
| 7 | 19 | 31 | 43 | 4 | 6 | 8 | 10 |
| 10 | 24 | 38 | 52 | 7 | 10 | 13 | 16 |
| 13 | 29 | 45 | 61 | 10 | 14 | 18 | 22 |
| 16 | 34 | 52 | 70 | 1 | 8 | 15 | 22 |
| 19 | 39 | 59 | 79 | 4 | 8 | 12 | 16 |
| 22 | 44 | 66 | 88 | 7 | 8 | 9 | 10 |
| 25 | 49 | 73 | 97 | 10 | 8 | 6 | 4 |

(d)

**Figure 2.** An example using the STC method (a) The original $8 \times 8$ image. (b) The partitioned homogeneous blocks of (a). (c) The bintree representation of (b). (d) The estimated image of (a).

where

$$g_5 = 4 + \frac{4-4}{7-0}(1-0) = 4$$

and

$$g_6 = 25 + \frac{4-25}{7-0}(1-0) = 22$$

The absolute difference between $g(1,0)$ and $g_{est}(1,0)$ is

$$|g(1,0) - g_{est}(1,0)| = |14-4| = 10$$

Since the value 10 is larger than the specified error tolerance $\varepsilon = 5$, it violates $\varepsilon = 5$. Therefore, Figure 2(a) is subdivided into two equal-sized subimages by cutting the $x$-axis. By the same argument, the resulting partitioned homogeneous blocks are denoted by $B_0$, $B_1$, and $B_2$ and depicted in Figure 2(b). The corresponding bintree representation is illustrated in Figure 2(c). The four corners of the homogeneous block $B_0$ have grey levels $g_1(0,0)=4$, $g_2(3,0)=34$, $g_3(0,7)=25$, and $g_4(3,7)=97$. By Eqn (2), the estimated grey-level of the pixel at (0,1) is calculated by

$$g_{est}(0,1) = 4 + \frac{25-4}{7-0}(1-0) = 7$$

where

$$g_5 = 4 + \frac{34-4}{3-0}(0-0) = 4$$

and

$$g_6 = 25 + \frac{97-25}{3-0}(0-0) = 25$$

Similarly, we have $g_{est}(0,2)=10$, $g_{est}(0,3)=13$, ..., and $g_{est}(3,6)=88$. Finally, the estimated image of Figure 2(a) is shown in Figure 2(d), which depicts all the corresponding estimated grey-levels obeying $\varepsilon = 5$. From Figure 2(c) and (d), the corresponding S-tree representation is listed below:

linear-tree table: 0 1 0 1 1

color table: $(B_{0g1}, B_{0g2}, B_{0g3}, B_{0g4})$, $(B_{1g1}, B_{1g2}, B_{1g3}, B_{1g4})$, $(B_{2g1}, B_{2B2}, B_{2g3}, B_{2g4}) = (4,34,25,97)$, $(1,22,10,4)$, $(1,4,10,22)$

In the above S-tree representation, there are four entities in the colour table, where each entity contains four grey-levels. The binary string 01011 in the linear-tree table is used to capture the geometrical relationship among these homogeneous blocks.

## Computing Moments on Compressed Images

Let $I = \{(x, y)\}|0 \leq x \leq N-1, 0 \leq y \leq N-1\}$ be the image domain, decomposed into a set of $K$ homogeneous blocks using the STC method mentioned above. Let $\{B_i|i = 0, 1, \ldots, K-1\}$ denote the set of these $K$ homogeneous blocks. Following the notations used in Figure 1, $B_i$ is represented by

$$B_i = \{(x, y)|x_1^{(i)} \leq x \leq x_2^{(i)}, y_1^{(i)} \leq y \leq y_2^{(i)}\}$$

where $B_i \cap B_j = \phi$ for $i \neq j$ and $I = \bigcup_{i=0}^{K-1} B_i$.

From Eqns (1) and (2), the computation of $m_{pq}$ is given by

$$m_{pq} = \sum_{x=0}^{N=1} x^p \sum_{y=0}^{N-1} y^q g_{est}(x, y) \tag{3}$$

It takes $O(N^2)$ time to compute the moments based on Eqn (3) directly. In what follows, we need to reduce the time requirement from $O(N^2)$ to $O(N\sqrt{K})$, commonly $K < N^2$ due to the compression effect (see section, Experimental Results). For each $x$, let $I_x = \{(x, y) \ |0 \leq y \leq N-1\}$ and we have

$$I_x = I \cap I_x = \bigcup_{i=0}^{k-1} (B_i \cap I_x)$$

Since $(B_i \cap I_x) \cap (B_j \cap I_x) = \phi$ for $i \neq j$, we further have

$$\begin{aligned}\sum_{y=0}^{N-1} y^q g_{est}(x, y) &= \sum_{(x,\, y) \in I_x} y^q g_{est}(x, y) \\ &= \sum_{(x,\, y) \in \bigcup_{i=0}^{K-1} (B_i \cap I_x)} y^q g_{est}(x, y) \\ &= \sum_{i=0}^{K-1} \sum_{(x,\, y) \in B_i \cap I_x} y^q g_{est}(x, y) \end{aligned} \tag{4}$$

Let $v_q^{(B_i)}(x) = \sum_{(x,\, y) \in B_i \cap I_x} y^q g_{est}(x, y) = \sum_{y=y_1^{(i)}}^{y_2^{(i)}} y^q g_{est}(x, y)$ and

$$r_q(x) = \sum_{y=0}^{N-1} y^q g_{est}(x, y) = \sum_{i=0}^{K-1} v_q^{(B_i)}(x)$$

$$\text{for} \quad x = 0, 1, \cdots, N-1$$

The following lemma shows that the computation of $v_q^{(B_i)}(x)$ can be reduced from $O(y_2^{(i)} - y_1^{(i)})$ time to $O(1)$ time.

*Lemma 1.* $v_q^{(B_i)}(x) = \sum_{y=y_1^{(i)}}^{y_2^{(i)}} y^q g_{est}(x, y)$ for $q = 0, 1, 2,$ and 3 that can be computed in $O(1)$ time.

*Proof*. Consider the pixels on the top boundary and the bottom boundary at positions $(x, y_1)$ and $(x, y_2)$, respectively, for $x_1 \leq x \leq x_2$. By Eqn (2), we have

$$g_{est}(x, y_1) = g_1 + \frac{g_2 - g_1}{x_2 - x_1}(x - x_1)$$

and

$$g_{est}(x, y_2) = g_3 + \frac{g_4 - g_3}{x_2 - x_1}(x - x_1)$$

For simplifying the notations used, let

$$D_1 = g_1 + \frac{g_2 - g_1}{x_2 - x_1}(x - x_1)$$

and

$$\begin{aligned} D_2 &= \frac{(g_3 + \frac{g_4 - g_3}{x_2 - x_1}(x - x_1)) - (g_1 + \frac{g_2 - g_1}{x_2 - x_1}(x - x_1))}{y_2 - y_1} \\ &= \frac{g_3 - g_1 + \frac{g_4 - g_3 - g_2 + g_1}{x_2 - x_1}(x - x_1)}{y_2 - y_1} \\ &= \frac{g_3 - g_1}{y_2 - y_1} + \frac{g_4 - g_3 - g_2 + g_1}{(y_2 - y_1)(x_2 - x_1)}(x - x_1) \end{aligned}$$

then we have $g_{est}(x, y) = D_1 + (y - y_1) \times D_2$, where the values of $D_1$ and $D_2$ are dependent on the values of $x$. The computation of $v_q^{(B_i)}(x) = \sum_{y=y_1}^{y_2} y^q g_{est}(x, y)$ can be rewritten as

$$\begin{aligned} v_q^{(B_i)}(x) &= \sum_{y=y_1}^{y_2} y^q g_{est}(x, y) \\ &= \sum_{y=y_1}^{y_2} y^q (D_1 + (y - y_1) \times D_2) \\ &= (D_1 - y_1 D_2) \sum_{y=y_1}^{y_2} y^q + D_2 \sum_{y=y_1}^{y_2} y^{q+1} \end{aligned}$$

The following four equalities are well-known and they will be used later:

$$\sum_{y=0}^{N-1} y = \frac{N(N-1)}{2},$$

$$\sum_{y=0}^{N-1} y^2 = \frac{N(N-1)(2N-1)}{6}$$

$$\sum_{y=0}^{N-1} y^3 = \frac{N^2(N-1)^2}{4} \quad \text{and}$$

$$\sum_{y=0}^{N-1} y^4 = \frac{N(N-1)(2N-1)(3N^2-3N-1)}{30}$$

The summation term $\sum_{y=y_1}^{y_2} y^q$ for $q=0$, 1, 2, and 3 can be calculated directly by the following formulas:

$$\sum_{y=y_1}^{y_2} y = \sum_{y=0}^{y_2} y - \sum_{y=0}^{y_1-1} y = \frac{y_2(y_2+1) - y_1(y_1-1)}{2}$$

$$\sum_{y=y_1}^{y_2} y^2 = \sum_{y=0}^{y_2} y^2 - \sum_{y=0}^{y_1-1} y^2 = \frac{y_2(y_2+1)(2y_2+1) - y_1(y_1-1)(2y_1-1)}{6}$$

$$\sum_{y=y_1}^{y_2} y^3 = \sum_{y=0}^{y_2} y^3 - \sum_{y=0}^{y_1-1} y^3 = \frac{y_2^2(y_2+1)^2 - y_1^2(y_1-1)^2}{4}$$

and

$$\sum_{y=y_1}^{y_2} y^4 = \sum_{y=0}^{y_2} y^4 - \sum_{y=0}^{y_1-1} y^4 = \frac{y_2(y_2+1)(2y_2+1)(3y_2^2+3y_2-1) - y_1(y_1-1)(2y_1-1)(3y_1^2-3y_1-1)}{30}$$

For $v_q^{(B_i)}(x)$, $0 \le q \le 3$, $v_0^{(B_i)}(x)$, $v_1^{(B_i)}(x)$, $v_2^{(B_i)}(x)$, and $v_3^{(B_i)}(x)$ can be computed by

$$v_0^{(B_i)}(x) = (D_1 - y_1 D_2) \times (y_2 - y_1 + 1) + D_2 \times \frac{y_2(y_2+1) - y_1(y_1-1)}{2}$$

$$v_1^{(B_i)}(x) = (D_1 - y_1 D_2) \times \frac{y_2(y_2+1) - y_1(y_1-1)}{2} + D_2 \times \frac{y_2(y_2+1)(2y_2+1) - y_1(y_1-1)(2y_1-1)}{6}$$

$$v_2^{(B_i)}(x) = (D_1 - y_1 D_2) \times \frac{y_2(y_2+1)(2y_2+1) - y_1(y_1-1)(2y_1-1)}{6} + D_2 \times \frac{y_2^2(y_2+1)^2 - y_1^2(y_1-1)^2}{4},$$

and

$$v_3^{(B_i)}(x) = (D_1 - y_1 D_2) \times \frac{y_2^2(y_2+1)^2 - y_1^2(y_1-1)^2}{4} + D_2 \times \frac{y_2(y_2+1)(2y_2+1)(3y_2^2+3y_2-1) - y_1(y_1-1)(2y_1-1)(3y_1^2-3y_1-1)}{30},$$

It is clear that $v_q^{(B_i)}(x)$, $0 \le q \le 3$, can be computed in $O(1)$ time since each $v_q^{(B_i)}(x)$ only needs a few arithmetic operations. We complete the proof.

After considering one block case $B_i$, let us look at the right-hand side of Eqn (4). Totally, there are $K$ blocks to be considered. For exposition, let us return to Figure 2. There are three blocks, $B_0$, $B_1$, and $B_2$ to be considered. From Figure 2(a) and Eqn (3), we have $m_{pq} = \sum_{x=0}^{7} x^p \sum_{y=0}^{7} y^q g_{est}(x,y)$.

We first consider the first interval $0 \le x \le 3$. For $x=0$, we only consider $B_0$ due to $I_0 \cap B_0 \ne \phi$, but $I_0 \cap B_1 = \phi$ and $I_0 \cap B_2 = \phi$. By Lemma 1, it takes $O(1)$ time for computing $\sum_{y=0}^{N-1} y^q g_{est}(0,y) = \sum_{y=0}^{7} y^q g_{est}(0,y)$. By the same arguments, for $x=1$ with respect to $I_1$, it takes $O(1)$ time for computing $\sum_{y=0}^{7} y^q g_{est}(1,y)$, and so on. Combining the total time required for $I_0$, $I_1$, $I_2$, and $I_3$, it takes $O(x_2^{(0)} - x_1^{(0)})$ time for computing $\sum_{x=0}^{3} x^p \sum_{y=0}^{7} y^q g_{est}(x,y)$.

We next consider the remaining interval $4 \le x \le 7$. In this interval, we only consider $B_1$ and $B_2$ due to $I_x \cap B_0 = \phi$, but $I_x \cap B_1 \ne \phi$ and $I_x \cap B_2 \ne \phi$. By the same arguments discussed in the last paragraph, it takes $O((x_2^{(1)} - x_1^{(1)}) + (x_2^{(2)} - x_1^{(2)}))$ time for computing $\sum_{x=4}^{7} x^p \sum_{y=0}^{7} y^q g_{est}(x,y)$. Consequently, for the whole interval $0 \le x \le 7$, it takes $O((x_2^{(0)} - x_1^{(0)}) + (x_2^{(1)} - x_1^{(1)}) + (x_2^{(2)} - x_1^{(2)}))$ time for computing $\sum_{x=0}^{7} x^p \sum_{y=0}^{7} y^q g_{est}(x,y)$.

In the S-tree representation mentioned above, if the number of leaves in the S-tree is $K$, then it means that the number of homogeneous blocks is $K$ for the $N \times N = (2^n \times 2^n)$ image. Among these $K$ blocks, let the number of squared blocks be $k_1$ and the number of rectangular blocks be $k_2$ such that $K = k_1 + k_2$. For convenience, let these $k_1$-squared blocks be of sizes $(s_1 \times s_1)$, $(s_2 \times s_2)$, $\ldots$, and $(s_{k_1} \times s_{k_1})$, where $s_i = 2^{l_i}$, $1 \le i \le k_1$ and $1 \le l_i \le n$; let these $k_2$ rectangular blocks be of sizes $(2r_1 \times r_1)$, $(2r_2 \times r_2)$, $\ldots$, and $(2r_{k_2} \times r_{k_2})$, where $r_j = 2^{m_j}$, $1 \le j \le k_2$ and $1 \le m_j \le n-1$.

From Lemma 1 and the above description, we have the following result.

*Lemma 2*. For each squared homogeneous block with size $s_i \times s_i$, the time complexity required in the computation of $m_{pq}$ is proportional to $s_i$; for each rectangular block with size $2r_j \times r_j$, the time complexity required in the computation of $m_{pq}$ is proportional to $r_j$.

We now require to analyze the total time complexity in the worst case for the proposed method. We have the main result.

*Theorem 3*. Given an $N \times N$ grey image, suppose it is compressed into a compressed image with $K$ blocks, then the computation of moments can be done in $O(N\sqrt{K})$ time.

*Proof*. From Lemma 2, it is known that the total time complexity is bounded by

$$T = \sum_{i=1}^{k_1} s_i + \sum_{j=1}^{k_2} r_j$$

Since the image size is of $N \times N$, we have $\sum_{i=1}^{k_1} s_i^2 + \sum_{j=1}^{k_2} 2r_j^2 = \sum_{i=1}^{k_1} s_i^2 + 2\sum_{j=1}^{k_2} r_j^2 = N^2$. Let $\vec{u}$ be a $K$-dimensional vector and $\vec{u} = (s_1, s_2, \ldots, s_{k_1}, \sqrt{2}r_1, \sqrt{2}r_2, \ldots, \sqrt{2}r_{k_2})$. In addition, let $\vec{v} = (\underbrace{1, 1, \ldots, 1}_{k_1}, \underbrace{1/\sqrt{2}, 1/\sqrt{2}, \ldots, 1/\sqrt{2}}_{k_2})$. By the Cauchy–Schwarz inequality [16], we have

$$|\vec{u} \cdot \vec{v}| \le \|\vec{u}\|_2 \times \|\vec{v}\|_2$$

From

$$\vec{u} \cdot \vec{v} = \sum_{i=1}^{k_1} s_i + \sum_{j=1}^{k_2} r_j$$

$$\|\vec{u}\|_2 = \sqrt{\sum_{i=1}^{k_1} s_i^2 + 2\sum_{j=1}^{k_2} r_j^2} = N$$

and

$$\|\vec{v}\|_2 = \sqrt{k_1 + \frac{k_2}{2}}$$

by Cauchy–Schwarz inequality, we have

$$T = \sum_{i=1}^{k_1} s_i + \sum_{j=1}^{k_2} r_j \le N \times \sqrt{k_1 + \frac{k_2}{2}} \le N \times \sqrt{k_1 + k_2}$$
$$= N \times \sqrt{K}$$

From the definition of the big–$O$ notation [15], we thus have $T = O(N\sqrt{K})$. We complete the proof.

By traversing the S-tree representation and from Theorem 1, the computation of moments $m_{00}$, $m_{10}$, $m_{20}$, $m_{30}$, $m_{01}$, $m_{11}$, $m_{21}$, $m_{02}$, and $m_{12}$ can be done in $O(N\sqrt{K})$ time.

## Experimental Results

In the STC method, a minimum block allowable contains four pixels, and for this case, no data compression is achieved in encoding the minimum block. In order to have a better robustness to noises and get a better compression ratio, we allow one pixel in a homogeneous block to exceed the specified error tolerance. This leads to one noise robustness for any homogeneous block. Three $512 \times 512$ grey images, Lena, F16, and Pepper are used to compare the performance among the proposed method, the indirect method (first decompressing the compressed image, then computing moments on the decompressed image), and the conventional method. Here, the conventional method computes the moments on the decompressed image directly. All the related implementations are performed using Borland C++ Builder 5.0 on the IBM compatible Celeron microprocessor with 450 MHz.

Given the error tolerance $\varepsilon = 21$, Table 1 lists the average bits per pixel (*BPP*), signal-to-noise ratios (*SNR*s) [9], the number of partitioned blocks ($K$), and the execution time in terms of seconds. The *SNR* is used to measure the similarity between the original image and

**Table 1.** Performance comparison

| Figure | *BPP* | *SNR* | *K* | Proposed method | Indirect method | Conventional method |
|---|---|---|---|---|---|---|
| Lena | 1.12 | 26.9 | 8676 | 0.010 | 0.241 | 0.090 |
| Pepper | 1.50 | 26.5 | 11680 | 0.013 | 0.238 | 0.090 |
| F16 | 1.67 | 30.8 | 13070 | 0.015 | 0.235 | 0.090 |

**Table 2.** Accuracy comparison

| | $m_{00}$ | $m_{10}$ | $m_{20}$ | $m_{30}$ | $m_{01}$ | $m_{11}$ | $m_{21}$ | $m_{02}$ | $m_{12}$ |
|---|---|---|---|---|---|---|---|---|---|
| Lena-O | 3.25E07 | 8.67E09 | 3.01E12 | 1.16E15 | 8.05E09 | 2.19E12 | 7.71E14 | 2.70E12 | 7.38E14 |
| Lena | 3.24E07 | 8.63E09 | 3.00E12 | 1.16E15 | 8.02E09 | 2.18E12 | 7.67E14 | 2.69E12 | 7.35E14 |
| F16-O | 4.66E07 | 1.22E10 | 4.26E12 | 1.66E15 | 1.17E10 | 3.04E12 | 1.06E15 | 4.00E12 | 1.02E15 |
| F16 | 4.65E07 | 1.22E10 | 4.24E12 | 1.65E15 | 1.17E10 | 3.04E12 | 1.06E15 | 3.99E12 | 1.02E15 |
| Pepper-O | 3.07E07 | 7.88E10 | 2.69E12 | 1.04E15 | 7.56E09 | 1.88E12 | 6.29E14 | 2.54E12 | 6.15E14 |
| Pepper | 3.07E07 | 7.88E10 | 2.69E12 | 1.04E15 | 7.56E09 | 1.88E12 | 6.30E14 | 2.54E12 | 6.16E14 |

the decompressed image and is defined by

$$SNR(\mathrm{dB}) = 10\log_{10}\frac{\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} g^2(x,y)}{\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\{g(x,y)-g_{est}(x,y)\}^2}$$

It is observed that the average executing time improvement ratio of the proposed method over the indirect method is

$$95\% = \frac{\text{Time (Indirect Method)} - \text{Time (Proposed Method)}}{\text{Time (Indirect Method)}} = \frac{0.238 - 0.014}{0.238}$$

the executing time improvement ratio of the proposed method over the conventional method is $86\% = (0.090-0.014)/0.90$.

Table 2 illustrates the calculated values of the concerning moments. In Table 2, the symbol Lena-O denotes the original Lena image without any distortion; the symbol Lena is the same as the compressed Lena image in Table 1. It is observed that the proposed method has a high accuracy when compared to the conventional method running on the original Lena image, i.e. Lena-O. For example, the calculated value of $m_{10}$ is $8.63 \times 10^9$ using the proposed method on the compressed image; the calculated value of $m_{10}$ is $8.67 \times 10^9$ using the conventional method on Lena-O. For this case, the relative error is about 0.46% which is infinitesimal.

In summary, experimental results reveal a significant computational advantage of the proposed algorithm while preserving a high accuracy of moments and good compression ratio.

## Conclusions

We have presented an efficient algorithm for computing low-order moments in $O(N\sqrt{K})$ time. The detailed time complexity analysis is also given. Three real images have been used to test the performance comparison among the proposed method, the indirect method, and the conventional method. Experimental results reveal a significant computational advantage of the proposed algorithm while preserving a high accuracy of moments and good compression ratio. The results of this paper extend the previous results by Spiliotis and Mertzios [7] from the binary image domain to the grey image domain. The question as to how to plug the refined moment calculation technique [17] into our computational method is an interesting research issue.

## References


1. Hu, M.K. (1962) Visual Pattern Recognition by Moment Invariants. *IEEE Transactions on Information Theory* **8**: 179–187.
2. Pei, S.C. & Liou, L.G. (1994) Using Moments to Acquire the Motion Parameters of a Deformable Object Without Correspondences. *Image and Vision Computing* **12**: 475–485.
3. Pei, S.C. & Horng, J.H. (1999) A Moment-based Approach for Deskewing Rotationally Symmetric Shapes. *IEEE Transactions on Image Processing* **8**: 1831–1834.
4. Sonka, M., Hlavac, V. & Boyle, R. (1998) *Image Processing, Analysis, and Machine Vision* (2nd edn). New York: PWS.
5. Tsai, W.H. (1985) Moment-preserving Thresholding: a New Approach. *Computer Vision, Graphics, and Image Processing* **29**: 377–393.
6. Yang, C.K., Lin, J.C. & Tsai, W.H. (1997) Color Image Compression by Moment-preserving and Block Truncation Coding Techniques. *IEEE Transactions on Communications* **45**: 1513–1516.
7. Spiliotis, I.M. & Mertzios, B.G. (1998) Real-time Computation of Two-dimensional Moments on Binary Images using Image Block Representation. *IEEE Transactions on Image Processing* **7**: 1609–1615.
8. Distasi, R., Nappi, M. & Vitulano, S. (1997) Image compression by B-tree Triangular Coding. *IEEE Transactions on Communications* **45**: 1095–1100.
9. Pennebaker, W.B. & Mitchell, J.L. (1993) *JPEG: Still Image Data Compression Standard*. New York.
10. Jonge, W.D., Scheuermann, P. & Schijf, A. (1994) $S^+$–Trees: An Efficient Structure for the Representation

of Large Pictures. *Computer Vision and Image Understanding* **59**: 265–280.
11. Foley, J.D., Dam, A.V., Feiner, S.K. & Hughes, J.F. (1990) *Computer Graphics*, *Principle*, *and Practice* (2nd edn). Reading, MA: Addison-Wesley.
12. Chung, K.L. & Wu, J.G. (2000) Improved Image Compression using S-tree and Shading Approach. *IEEE Transactions on Communications* **48**: 748–751.
13. Samet, H. (1990) *The Design and Analysis of Spatial Data Structures*. New York: Addison-Wesley.
14. Samet, H. (1990) *Applications of Spatial Data Structures*. New York: Addison-Wesley.
15. Cormen, T.H., Leiserson, C.E. & Rivest, R.L. (1990) *Introduction to Algorithms*. Cambridge, MA: The MIT Press.
16. Hoffman, K. & Kune, R. (1971) *Linear Algebra* (2nd edn). New Jersey: Prentice-Hall.
17. Flusser, J. (2001) Refined Moment Calculation using Image Block Represenation. *IEEE Transactions on Image Processing* **9**: 1977–1978.