ADAPTIVE TWO-PHASE TREE-BASED SPATIAL DATA STRUCTURES AND THEIR APPLICATION TO IMAGE COMPRESSION

Kuo-Liang Chung[a], Shou-Yi Tseng[b], and I-Chien Chen[a]

[a] Department of Information Management Institute of Computer Science & Information Engineering
National Taiwan University of Science and Technology , Taipei, Taiwan, R.O.C.
klchung@cs.ntust.edu.tw
[b] Department of Computer & Information Science, Soochow University, Taipei, Taiwan, R.O.C.
tseng@cis.scu.edu.tw

ABSTRACT

Given a binary image, this paper first observes that the conventional tree-based spatial data structures (SDSs) spend some amount of memory for storing the external nodes and internal nodes near leaves. Next, we present an adaptive two-phase (ATP) representation to reduce the memory requirement. In addition, some geometric operations, such as computing the area and centroid, on the proposed ATP representation are also discussed. Experimental results show that not only the proposed ATP representation can reduce the memory requirement when compared to the existing ones, but also can lead to faster concerning geometrical operations. Finally, the proposed results are extended to compress gray images.

1. INTRODUCTION

Quadtree is the best-known spatial data structure (SDS) for representing the binary image and can reduce the memory requirement through the use of aggregation of homogeneous blocks [8]. Based on different kinds of SDSs, Samet [7] gave an excellent survey on many applications in computer graphics, image processing, geographic information systems, image database, pattern recognition, etc.

In order to reduce the storage requirement, some space-saving representations have been proposed. Gargantini [3] presented a pointerless SDS, called the linear quadtree. The DF-expression [6] encodes each node of the quadtree in depth first search (DFS) manner using the symbol 'G','B', or 'W' to indicate the gray node, black node or white node, respectively. Based on the bintree, Jonge *et al*. [5] presented the S-tree representation. Recently, Distasi *et al.* [2] presented an efficient bintree triangular coding (BTTC) method for compressing gray images and the method has shorter execution time when compared to the JPEG [10], although the bit rates are higher by a factor of about 2. Based on the modified S-tree data structure and the Gouraud shading method, Chung and Wu [1] presented the S-Tree Compression (STC) method which improves the execution time of the BTTC method in the ratio less than 1/2 while preserving the same image quality and bits rates.

In this paper, we first observe that these conventional SDSs mentioned above spend some amount of memory for storing external nodes and internal nodes near leaves, especially the image with some detailed texture. In order to reduce the memory requirement, we present the adaptive two-phase (ATP) representation to improve the conventional tree-based SDSs, and it leads to better compression performance on the most memory consuming part. In the first phase, we follow the conventional tree-based SDS from root to a specific level, say *s*, of the tree. There are three kinds of leaves at level *s* representing the entirely white subimages, the entirely black subimages, and the gray subimages. The third kind of leaf that we name it as the gray leaf is the memory consuming part in the conventional SDSs. In the second phase, each gray leaf obtained from the first phase is coded by the connected component string (CCS) coding scheme. The CCS is obtained by employing the morphological technique [9] in the connected component analysis and can efficiently represent the gray leaf. Experimental results show that the proposed ATP representation can reduce the memory requirement from 8% to 71% when compared to the linear quadtree [3], DF-expression [6], and S-tree representation [5]. The effect of memory reduction also leads to better computing performance for the concerning geometric operations such as computing the area and the centroid. Besides the binary image, we extend the proposed ATP representation to compress gray images. Experimental results reveal that the proposed compression method for gray images has better compression performance when compared to the recent result [1] while preserving the same image quality.

2. CONVENTIONAL TREE-BASED SDSs

In this section, we take a simple example to introduce three existing tree-based SDSs. For a quadtree, if the image is entirely black (white), then the root node is

labeled with 1 (0). Otherwise, the root node is further divided into four equal-sized subimages. For the quadrants, they are labeled *sw* (southwest), *se* (southeast), *nw*(northwest), and *ne*(northeast), respectively. The quadtree decomposition is based on repeatedly subdividing the subimages until the subimage is entirely black or white. Given a binary image with *8x8* as shown in Fig. 1, the corresponding quadtree is shown in Fig. 2.

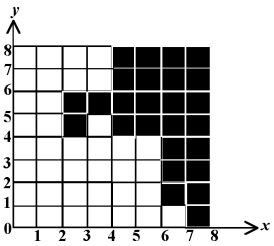


Fig. 1. A binary image example.

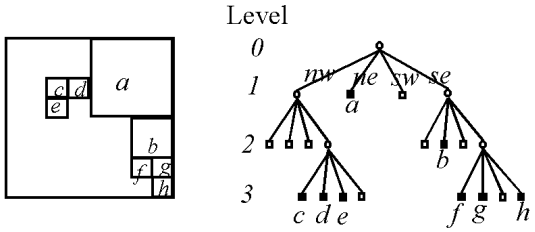


Fig. 2. The quadtree representation of Fig. 1.

2.1 Linear Quadtree

Without using pointers, a linear quadtree [6] represents the quadtree by a set of codes, and each code is obtained by encoding a path from the root to the black node, i.e. external node, in the quadtree. Let the *sw* quadrant, the *se* quadrant, the *nw* quadrant, and the *ne* quadrant be encoded with 0, 1, 2, and 3, respectively. Fig. 3 illustrates the assigned codes on each quadrant. The node *c* in Fig. 3 is encoded by 030; the node *d* is encoded by 031, and the node *a* is encoded by *1X*, where *X* is a don't-care symbol. The don't-care symbol '*X*' denotes this node is not at the bottom level of the tree and its code is ended at '*X*'. By traversing the quadtree in depth-first search (DFS) manner to code the black nodes, the quadtree in Fig. 3 is coded as 030 031 032 1X 31X 330 331 333.

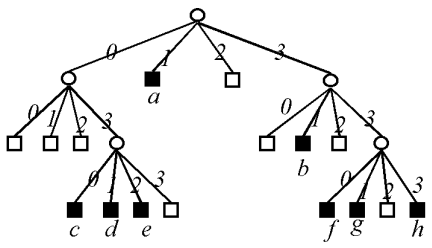


Fig. 3 The assigned digits of the linear quadtree.

2.2 DF-expression

Given a quadtree, the DF-expression [6] is obtained by traversing the quadtree in DFS manner. The DF-expression is a sequence consisting of three symbols '*G*','*W*', and '*B*', which denote the gray node, white node, and black node, respectively. During the traversal, if a gray node is encountered, the symbol '*G*' is appended to the DF-expression; if a white node is encountered, the symbol '*W*' is appended to the DF-expression, and the symbol '*B*' is appended if a black node is encountered. For example, the DF-expression of Fig. 2 is *GGWWWGBBBWBWGWBWGBBWB*.

2.3 S-tree

The S-tree is based on the bintree [8] structure. Given an image, the corresponding bintree is obtained by recursively subdividing the image into two equal-sized subimages until the subimage is totally black or white. At each step, the partition is alternated between the *x*- and *y*- axes. The corresponding bintree of Fig. 1 is shown in the right side of Fig. 4 and the partitioned subimages are shown in the left side of Fig. 4.

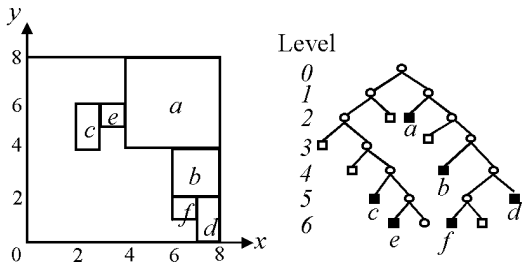


Fig. 4. The bintree representation of Fig. 1.

The S-tree representation is obtained by traversing the bintree in breadth-first search (BFS) manner and the traversed result is stored in two array tables, namely, the linear-tree table and the color table. While traversing the bintree, a '1' ('0') is emitted to the linear-tree table when an external (internal) node is encountered. Meanwhile, a '1' ('0') is emitted to the color table when a black (white) leaf node is encountered. For example, the S-tree representation for Fig. 4 is listed below:

linear-tree table: 0 00 0110 1010 1010 1001 1111

color table: 010001111010

3. PROPOSED ADAPTIVE TWO-PHASE (ATP) REPRESENTATION

In this section, we first describe the idea of the proposed ATP representation. Second, how to apply the morphological dilation operator to the connected component analysis is introduced. Then, the CCS coding scheme for representing the connect components is described.

3.1 The Idea of the Proposed ATP representation

In the first phase, we employ one of the conventional SDSs to represent an image. Instead of subdividing each subimage into an entirely black or white subimage, we stop the tree decomposition to a specified level to obtain an approximate quadtree (bintree) for the image. Fig. 5 is an example of the approximate quadtree that is obtained by stopping the decomposition at level 1. The approximate tree has three kinds of leaves, namely, the black leaves, the white leaves, and the gray leaves. Each black leaf represents an entirely black subimage; each white leaf represents an entirely white subimage, and each gray leaf represents a gray subimage. The gray leaf should be a subtree in the conventional SDS. As described before, the gray leaves are the memory consuming part in the conventional SDSs. To reduce the memory requirement, in the second phase, we code the gray leaves using the CCS coding scheme. A CCS is a bit stream which can efficiently represent a connected component in a subimage. The CCSs are obtained by employing the morphological technique in the connected component analysis for the subimages represented by gray leaves.

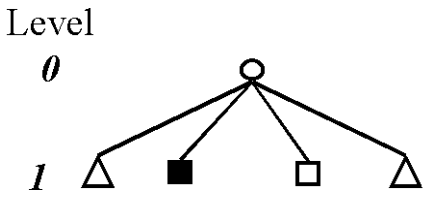


Fig. 5 The approximate tree of Fig. 1.

For example, the binary image shown in Fig. 1 should be represented by the quadtree in Fig. 2 using the conventional SDSs. In stead of decomposition the tree to level 3, we stop the decomposition at level 1 and obtain the approximate quadtree as in Fig. 5. The nodes marked as Δ'*s* in Fig. 5 are the gray leaves those will be coded using the CCS coding scheme in the second phase.

3.2 Connected Component Analysis

To find a connected component for the subimage represented by a gray leaf, we employ the morphological dilation operation [4,9]. Consider a set A to which the dilation operation will be associated with by the structuring element B. Let $\oplus$ denote the morphological dilation operator. The dilated set $A \oplus B$ is defined to be the union of all pixels under the support of the structuring element B. An example of the dilation operation is shown in Fig. 6 where each square box in A and B is a pixel of the binary image.

Let Y represent a connected component contained in the subimage G. Scanning G in a raster manner, we get a starting point of Y, say p. Then the following iterative expression yields all the pixels of Y:

$$X_k=(X_{k-1} \oplus B) \cap G \text{ for } k=1,2,3,\ldots \quad (1)$$

where $X_0 = p$, and $\cap$ is the pixel-wise intersection. For each iteration, X_{k-1} extends to its connected neighbors by the shape of structuring element B within the subimage G. The iterations will stop at X_n when all the neighbors of X_n within the G are visited. Here, the set X_n is indeed the connected component Y contained in the subimage G. Since there may be more than one connected component in the subimage G, we can easily continue the raster scanning to find the starting point of the next connected component. Each connected component in the subimage G is represented by a starting point following by a connected component string (CCS). The next subsection describe how to generate the bit stream CCS from Eq. (1).

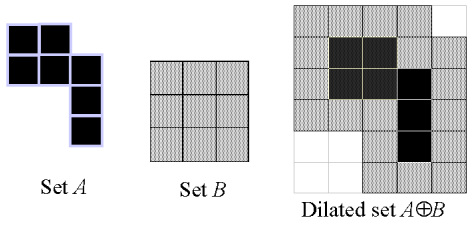


Fig. 6. The morphological dilation operation.

3.3 The CCS coding scheme

The subimage G is corresponding to one gray leaf on the approximate tree. For preserving better geometric connectivity, the structuring element B we used is 8-connected as shown in Fig. 6. When applying the dilation operation $X_{k-1} \oplus B$ in Eq. (1) for each pixel in X_{k-1}, there are eight neighbors of that pixel to be checked.

Initially, the set X_0 contains only one pixel, say p. To find its connected pixels, we start from the east neighbor of p and go clockwise to visit its eight neighbors. On visiting one neighbor, if this neighbor is not a black pixel, that means it is not a connected pixel, then we emit the bit '0' to the CCS. On the other hand, if this neighbor is a black pixel, that means it is a connected pixel, then the bit '1' is emitted to the CCS and then recursively go through the eight neighbors of this connected pixel.

An example of CCS coding simulation is illustrated in Fig. 7. In Fig. 7, the subimage G is of size *4 x 4* and its CCS is obtained by using five iterations. It is observed that the CCS needs only 11 bits in addition 4 bits for the location of starting point, so totally 15 bits are required in this example. If this subimage is represented by a quadtree, there are 10 leaf nodes to be encoded. No matter how efficient the conventional tree-based SDS is, the memory required in these leaf nodes is relatively larger than that of the CCS coding scheme. This is the main reason why the proposed ATP method can improve the conventional tree-based SDSs.

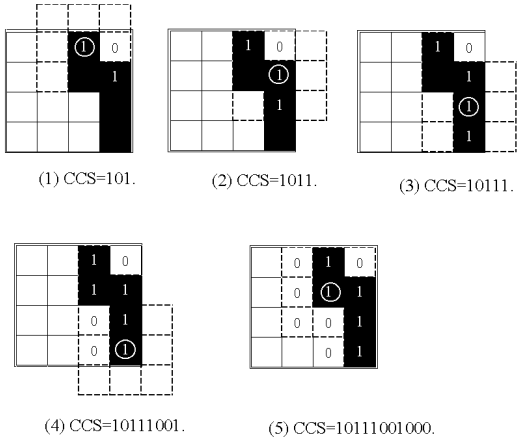


Fig. 7. CCS coding example.

As described above, the proposed ATP representation follows the conventional SDSs in the first phase to a specific tree level, then in the second phase, the CCS coding scheme is applied and has the memory-saving advantage when compared to the conventional tree-based SDSs. However, the concerning dilation operation in our method is time consuming for small subimages, such as for size *2x2* and *2x1*. In order to improve the encoding and decoding time, the bit pattern of such a small subimage is recorded directly. For example, in Fig. 2, the most right subtree at level 2 represents a subimage of size *2x2*. This small subimage has three black pixels, e.g. *f*, *g*, and *h*, and one white pixel. We record the bit pattern `1101' instead of iteratively performing the dilation operation. This improvement leads to faster encoding, decoding, and the concerning geometric operations.

4. GEOMETRIC OPERATIONS

In this section, the geometric operations for calculation the area and centroid on the ATP representation are described. Since the first phase is the same as the conventional SDSs, we only discuss the remaining operations for the CCS coded subimage. Each CCS coded subimage, say *G*, is corresponding to a gray leaf in the approximate tree from the first phase.

4.1 Area

The area of a binary image is defined as the number of black pixels of the whole image. As described in Section 3, we emit a bit '1' whenever a black pixel is encountered in encoding the CCS. Therefore, the number of '1's in a CCS is the area of its corresponding connected component. Given a CCS of the subimage *G*, we can easily obtain the area of *G* by counting the number of '1's in the CCS. If there exists any other connected components in the subimage, the area of each connected component can be computed from its CCS and they are added together to obtain the area of the subimage *G*.

For example, the subimage in Fig. 7, its CCS is '10111001000'. The number of '1's in the CCS is 5, so the area of this subimage is 5.

4.2 Centroid

Assume a binary image is of size $2^N x 2^N$. The origin of the image is at the top-left corner and the coordinate of the bottom-right pixel is (2^N-1, 2^N-1). The centroid is defined by

$$\begin{cases} \overline{X} = \dfrac{\sum x_i}{A} \\ \overline{Y} = \dfrac{\sum y_i}{A} \end{cases}$$

where (x_i,y_i), is the coordinate of each black pixel and A is the area of the given image.

Given a CCS, the recursive decoding procedure as describe in last section must be performed first to get the coordinate of each black pixel. In addition, the action of summation *x*-coordinate and summation *y*-coordinate is performed to obtain the subtotal of all the *x* and all the *y* of the subimage *G*.

5. EXPERIMENTAL RESULTS

In this section, some experiments are included to demonstrate the compression ratio among the proposed ATP representation, the linear quadtree (LQ), the DF-expression, and the S-tree. Three real binary images, namely, the butterfly, the floodmap, and the cup are used and shown in Fig. 8 (a), (b), and (c), respectively. Each image is of size *256 x 256* and requires 65536 bits. The experiments are performed on the IBM compatible personal computer Pentium III microprocessor with 500 MHz and 128MB RAM. The operation system is MS-Windows 98 and the program developing environment is Borland C++ Builder 4.0.

Fig. 8 Three testing real images.

Table 1 demonstrates the compression results on the three real images using the conventional DSDs and the proposed ATP representation. First, the image is encoded using the methods of linear quadtree (LQ), DF-expression, and S-tree representation. The conventional methods are one-phase when processing the image. Their compression results in bits are shown in the second column of Table 1. Applying the CCS to the gray leaves from the first phase at a specific level, a better compression ratio is obtained. The third column denotes the results in bits for the proposed ATP representation.

The size of the subimage corresponding to the gray leaf is shown in the next column. Finally, the improvement ratio is shown in the last column.

The height of the approximate tree decides the size of the CCS coded subimage. For example, a *256 x 256* image represented by a quadtree has nine levels in the one-phase LQ representation. Applying the ATP LQ representation, its approximate quadtree has six levels and the CCS coded subimage size is *8 x 8*. In other words, cutting the conventional LQ at the sixth level and coding the *8 x 8* subimage using CCS coding scheme can obtain the optimal compression ratio. That's why we list the CCS coded subimage size for every testing case.

Table 2 lists the computation time improvement for geometric operations using the same three testing images. The first column indicates the SDS method used; the second column shows the improvement for computing the area; and the third column shows the improvement for computing the centroid. The improvement ratio is computed by the time required by the conventional one-phase method (*ta*) subtracting from the time required by the proposed ATP representation (*tb*) and then divided by the value of *ta*, e.g. *(ta-tb)/ta*.The experimental results demonstrate that the proposed ATP representation can improve both the memory size and the computation time for geometric operations. It is observed that the more memory size is improved, the more computation time is improved.

6. APPLICATION TO COMPRESS GRAY IMAGES

Based on the modified S-tree data-structure and the Gourand shading method, the STC method [1] is an efficient bintree-based SDS for compressing gray images. In this section, we demonstrate the application to compress the gray images by applying the ATP representation to the STC method and it leads to a better compression ratio while preserving the image quality. In order to enhance the ATP compression effect, we apply the quadtree version of the STC method, namely the QSC.

6.1 The QSC Compression Method

In the QSC method, the original gray image is partitioned into several subimages based on the quadtree decomposition principle. For a subimage, 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 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 as:

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

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

$$g_6 = g_3 + (g_4 - g_3) \times \frac{x - x_1}{x_2 - x_1} \qquad (2)$$

Given a specified error tolerance ε, the image quality condition is

$$\left| g(x,y) - g_{est}(x,y) \right| \le \varepsilon. \qquad (3)$$

If all the pixels in the subimage, $x_2 \ge x \ge x_1$ and $y_2 \ge y \ge y_1$, hold the above quality condition, then this subimage is a leaf node in the quadtree, otherwise the subimage will be recursively divided into four equal sized square subimages until it holds the quality condition.

The QSC method represents the gray image using a linear-tree table and a color table like the S-tree mentioned in Section 2.3. The linear-tree table stores the geometric relationship of the divided subimages and the color table stores the graylevels on the four corner of all the leaf nodes.

6.2 The ATP QSC Representation

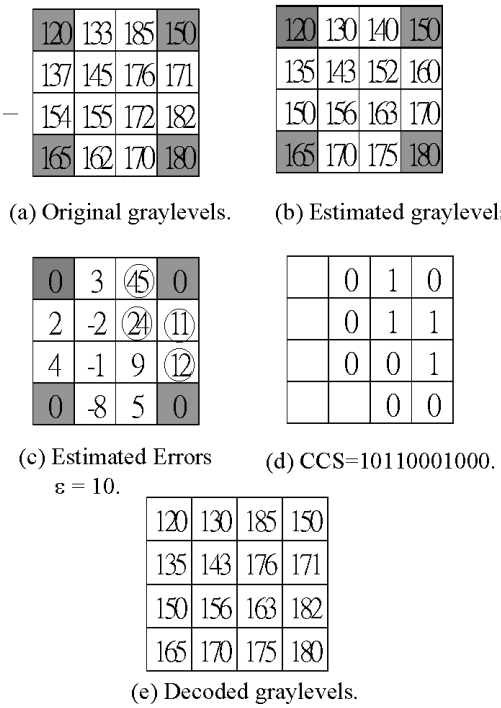
In the first phase, in stead of subdividing all the subimages into the ones each holding the image quality condition (see Eq. (3)), we stop the decomposition at a specified level to obtain an approximate quadtree for the image. In the second phase, the gray leaves those do not hold the image quality condition are coded using the CCS coding scheme.

To convert the CCS coding scheme from binary images to gray images, the pixel in the gray image to be coded to '1' in the CCS is determined by its estimated error. The estimated error is the difference between the the original graylevel and the estimated graylevel, e.g. $e(x,y) = g(x,y) - g_{est}(x,y)$. If the absolute value of $e(x,y)$ is greater than the specified error tolerance ε, then this pixel is coded by '1' in the CCS and the error value $e(x,y)$ is recorded in the color table. In other words, conceptually the pixel in the gray image with absolute estimated error greater than ε is treated as a black pixel in the binary image. Then, the encoding and decoding algorithm for CCS can be modified slightly to the gray images.

An example of the CCS coding for a *4 x 4* gray subimage is shown in Fig. 9. Fig. 9 (a) is the original subimage. Using the four corners' graylevels, those are marked by gray background in Fig. 9 (a), the estimated graylevels are computed by Eq. (2) and shown in Fig. 9 (b). The difference between the original graylevel and the estimated graylevel is the estimated errors $e(x,y)$ and is shown in Fig. 9 (c). Suppose ε is 10, then the pixels, each with absolute error over 10, are coded by CCS.

Consequently, the CCS is represented by 10110001000, and the corresponding graylevels recorded in the color table are 45, 11, 12, and 24.

On decoding a gray leaf, first the four corners' graylevels are obtained from the color table, then the estimated graylevels of the decoded subimage is computed by Eq. (2). The CCS for this subimage is decoded in the same way as CCS decoding algorithm described in Section 3.3. Whenever a bit '1' is decoded, the corresponding error value in the color table is added to its estimated graylevel. After decoding the whole CCS, we can obtain the decoded subimage as shown in Fig. 9 (e).



(a) Original graylevels. (b) Estimated graylevels.

(c) Estimated Errors ε = 10. (d) CCS=10110001000.

(e) Decoded graylevels.

Fig. 9 CCS coding for gray image.

6.3 The Experimental Results

We use four *512 x 512* gray images, Lena, Pepper, F16, and Baboo to compare the memory improvement of the original one-phase QSC representation and the proposed ATP QSC representation. Given the error tolerance ε, Table 3 is arranged in the same way as Table 1 except in terms of bytes. We observe that the compression improvement ratios ranges from 20% to 31% while preserving the same specified error tolerance $\varepsilon=10$. However, there is about 10% extra time to be paid.

7. CONCLUSION

We have presented the ATP representation to reduce the memory requirement required in the conventional tree-based SDSs. Experimental results show that the proposed ATP representation can reduce the memory requirement from 8% to 71% when compared to the linear quadtree [3], DF-expression [6], and S-tree representation [5]. Also, the proposed ATP can lead to faster concerning geometrical operations. In addition, we extend the proposed ATP representation to be a good candidate for compressing gray image which leads to wider applications for the tree-based SDSs.

8. REFERENCES

[1] K. L. Chung, and J. G. Wu, "Improved image compression using S-tree and shading approach," *IEEE Trans. On Communications*, Vol. 48, No. 5, pp. 748-751, 2000.

[2] R. Distasi, M. Nappi, and S. Vitulano, "Image compression by B-tree triangular coding," *IEEE Trans. on Communications*, Vol. 45, No. 9, pp. 1095-1100, 1997.

[3] I. Gargantini, "An effective way to represent quadtrees," *Communications of the ACM*, Vol. 25, No. 12, pp. 905-910, 1982.

[4] R. C. Gonzalez, and R. E. Woods, *Digital Image Processing*, Chapter 8, pp. 518-560, Addison Wesley, NewYork, 1992.

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

[6] E. Kawaguchi, and T. Endo, "On a method of binary picture representation and its application to data compression," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 2, No. 1, pp. 27-35, 1980.

[7] H. Samet, *Applications of Spatial Data Structures*, Addison-Wesley, New York, 1990.

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

[9] J. P. Serra, *Image Analysis and Mathematical Morphology*, Acdemic Press, NewYork, 1982.

[10] G. K. Wallace, "The JPEG still picture compression standard," *Communications of the ACM*, Vol. 34, No. 4, pp. 30-44, 1991.

Table 1. The Compressing Results.

	Method	One-phase Ma bits	ATP Mb bits	CCS coded Subimage size	Improvement ratio (Ma-Mb)/Ma
butterfly	LQ	42192	12558	8x8	70.24%
	DF	7582	6536	2x2	13.80%
	S-tree	7933	6889	1x2	13.16%
floodmap	LQ	18672	7819	8x8	58.12%
	DF	3767	3460	2x2	8.15%
	S-tree	4093	3717	1x2	9.18%
cup	LQ	38016	10848	8x8	71.46%
	DF	7443	6459	2x2	13.22%
	S-tree	7417	6649	1x2	10.35%

Table 2. The Computation Time Improvement for Geometric Operations.

	Method	Improvement ratio on area	Improvement ratio on centroid
butterfly	LQ	80.10%	35.04%
	DF	3.29%	18.76%
	S-tree	10.61%	22.42%
floodmap	LQ	70.53%	23.15%
	DF	8.22%	-15.08%
	S-tree	9.04%	34.07%
cup	LQ	80.67%	45.28%
	DF	7.53%	16.32%
	S-tree	8.19%	12.94%

Table 3. The Compression Results for Gray Images.

$\varepsilon=10$	One-phase QSC Ma bytes	ATP QSC Mb bytes	CCS coded subimage size	Improvement ratio (Ma-Mb)/Ma
Lena	140754	96924	4x4	31.14%
Pepper	164579	100568	8x8	38.89%
F16	111354	88456	4x4	20.56%
Baboo	254616	202587	8x8	20.43%