Two-Dimensional Template-Based Encoding for Linear Quadtree Representation*

Henry Ker-Chang Chang, Shing-Hua Liu, and Cheng-Kuan Tso

Abstract
A two-dimensional template-based encoding (2DTE) technique for linear quadtree construction is proposed. The 2DTE technique combines the concept of template mapping and the Morton sequence to encode regional data on an image. With the definition of the last homogeneous pixel, the new coding algorithm can be completed in a time linear to the number of pixels without repetitive scanning of image pixels. Compared with other linear quadtree coding methods, the proposed 2DTE technique has a linear n *time reduction of storage space if a 2^n by 2^n image is processed. The potential of the proposed 2DTE technique for encoding multicolored images is also described. Several empirical tests and theoretical analyses verify that the proposed 2DTE technique outperforms other linear quadtree construction algorithms. The proposed 2DTE technique is believed to be profitable for applications in image processing or geographic information systems.*

Introduction
The memory space required to store image data is large, and to store geographical data is even more significant. This requirement causes other problems for a geographic information system (GIS). As there are heavy loads for both geographical information processing and communication, much research is concerned with the storage problem, such as quadtree representation. A quadtree is a hierarchical data structure applied to represent geometric data. It is especially useful for raster-based image data processing and important for many applications, e.g., image processing, pattern recognition, computer vision, processing of spatial information, etc. Samet (1989) presented a detailed description of this situation. The hierarchical data structure of quadtree representation for regions or objects on an image makes storage more economical. The savings in memory therefore yield other advantages in data processing or communication. The original approach to represent geographical data is the quadtree data structure with pointers (Abel, 1984; Dyer *et al.*, 1980; Dyer, 1982; Hunter and Steiglitz, 1979; Klinger and Dyer, 1976; Klinger and Rhodes, 1979; Samet, 1980; Samet, 1981; Samet, 1984; Shaffer and Samet, 1987; Tanimoto and Paulidis, 1975). Recursive decomposition is applied to partition an image into four subimages represented by a tree structure with nodes of outdegree 4. Each subimage contains a square region of uniform intensity and is denoted as a node of a tree. A 2^n by 2^n image, n being the resolution factor for images, is recursively decomposed into four subimages until each quadrant has either all black pixels or all white pixels. Each node in the tree needs six fields to maintain the relationship with other nodes. There is one pointer for its father's node and there are four pointers to the nodes of four sons. The latter field holds the data type (color) of this node. Such a quadtree structure with pointers to represent regional objects has the merit that the original required memory space of 2^{2n} bytes for a raster image is significantly decreased. The memory space needed to represent the geographical objects on an image depends uniquely on the number of nodes in a quadtree. Samet (1989) proposed a corollary that the maximum number of nodes in a tree corresponding to an image is directly proportional to the resolution of the image; Hunter and Steiglitz (1979) proved that the quadtree corresponding to a polygon of perimeter p embedded in a 2^n by 2^n image has a maximum $24n - 19 + 24p$ nodes. Although a node in a tree may require several bits in order to be represented, the maximum memory space required will therefore have a complexity of $O(p+n)$, much less than the original memory complexity $O(2^{2n})$. Although the quadtree data structure yielded improved memory space, it is not the most efficient technique to represent geographical data in an image. Gargantini (1982a) showed that 66 percent of memory space can be further saved if a pointer-based quadtree is replaced by a linear coding method. This work motivated us to develop a data representation of a linear quadtree.

In this paper, we propose a linear quadtree representation to improve performance over the results derived in past research. The proposed approach, called Two-Dimensional Template-Based Encoding (2DTE), combines the idea of template matching and the Morton sequence to encode regional data on an image. The performance, in terms of memory space required and time complexities, for the proposed 2DTE technique is analyzed and compared. The experimental results verify our belief that the required memory for the proposed 2DTE technique provides a linear order of improvement over other linear quadtree techniques whereas the time complexity is still linear with respect to the number of pixels in an image.

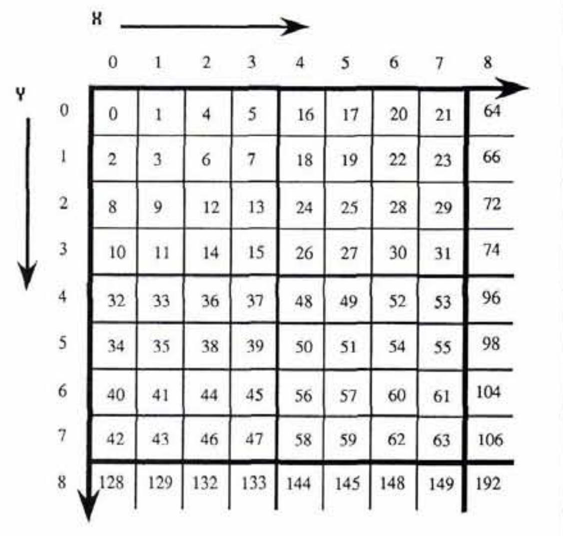
Review of the Linear Quadtree
Although the quadtree structure with pointers is attractive for many applications, several representation techniques without using pointers were developed to pursue the reduction of both memory space and execution time. Gargantini (1982a) provided an effective way to represent quadtrees, named linear quadtree. Her method was based on the Morton (1966) sequence. Only object areas, i.e., black pixels, in images are encoded and stored. Basically, a unique Morton

*Parts of this paper were presented at Signal Processing VI, Brussels, Belgium, 1992.

Graduate School of Resources Management, National Defense Management College, P.O. Box 90046-17, Chung-Ho, Taipei, Taiwan, R.O.C. (chang@rs590.ndmc.edu.tw).

Photogrammetric Engineering & Remote Sensing, Vol. 63, No. 11, November 1997, pp. 1275–1282.

0099-1112/97/6311-1275$3.00/0



Y \ X	0	1	2	3	4	5	6	7	8
0	0	1	4	5	16	17	20	21	64
1	2	3	6	7	18	19	22	23	66
2	8	9	12	13	24	25	28	29	72
3	10	11	14	15	26	27	30	31	74
4	32	33	36	37	48	49	52	53	96
5	34	35	38	39	50	51	54	55	98
6	40	41	44	45	56	57	60	61	104
7	42	43	46	47	58	59	62	63	106
8	128	129	132	133	144	145	148	149	192

Figure 1. Morton number assignment.

number is assigned for each region of uniform intensity. Four quadrants along a path from a parent node are visited and encoded as a weighted quaternary code — 0, 1, 2, and 3 — corresponding to the directions northwest, northeast, southwest, and southeast. Each black pixel may have more than one digit. The number of digits in each black pixel indicates how many times the decomposition is done. Thus, the weight 4^{n-h}, $1 \leq h \leq n$, of each digit identifies a situation in which the black pixel belongs to the hth decomposition. Once the image is scanned initially and quaternary codes are assigned to all black pixels, those quaternary codes are sorted and stored in an array or list. The quadtree is therefore linear because codes for each regional data on an image are recorded in a linear array. After all regional pixels are assigned quaternary codes, a condensation procedure is applied to merge neighboring pixels whose quaternary codes have the same representation except for the last digit. For each successful merge, four quaternary codes are eliminated from the list and a new quaternary code is appended with a special mark X generated to replace them. A final quadtree is derived until the condensation procedure is recursively applied to merge all neighboring regional pixels. All operations on the quadtree are then manipulated on the list or array rather than in a sequence following the order of pointers. Gargantini's representation technique has the advantage that overload of both memory and access duration for a tree structure with pointers is improved. Other related works (Gahegan, 1989; Gargantini, 1982b; Rosenfeld *et al.*, 1984; Wang, 1991) were based on Gargantini's coding technique. The major problem is that the requirement of memory size is proportional to the resolution of a given raster image. Her coding scheme may be further improved to diminish the requirement of memory space if the data structure is redesigned.

Lauzon *et al.* (1985) suggested another linear code for quadtree representation, Two-Dimensional Run-Encoding (2DRE). It combines runlength coding and Morton sequencing to construct a linear quadtree file. The structure is based on assignment of linear keys to each pixel in an image, and uses these keys to encode maximal leaves. After the image is given a Morton sequencing file, all consecutive records of the same color values, either black (B) or white (W), are deleted and represented by only the last record. The representative record has the final Morton number and its color value. The experimental results in their paper show that storage requirements for such a structure were decreased by pruning leaves based on value information. A 50 percent improvement was gained for the result of 2DRE compared with a file that records all maximal leaves with their associated color values. Methods to encode, decode, search, and overlay are discussed. Although the temporal complexity of the 2DRE method is still linear in the number of image resolution, the method of Lauzon *et al.* (1985) does not work as well as Gargantini's. We completed comparisons and found that the required memory space derived using the 2DRE method is larger than that using Gargantini's work.

Abel (1985) described another algorithm to construct a linear quadtree. His method resembles that designed by Gargantini (1982a) except that the assignment of Morton numbers to pixels follows the directional sequence SW, NW, SE, and NE. In addition, the linear keys for pixels are generated based on the radix 5. A compaction process is then developed to merge all neighboring keys of uniform intensity in order to diminish the required memory. The linear keys stored in the file impose actually a preorder traversal sequence on the quadtree. GIS operations, including union, intersection, and difference, are implemented.

Description of 2DTE

The proposed 2DTE technique, like a general quadtree decomposition, recursively decomposes an image into four subimages until each decomposed quaternary subdivision contains only uniform intensity. In the proposed 2DTE technique, each quadrant with a uniform intensity in an image is assigned a specific quaternary code consisting of four digits. Both object and background pixels on the image are encoded simultaneously. These quaternary codes are stored in a linear array to represent that image. The details of the proposed 2DTE technique, including both encoding and decoding algorithms, are explicitly described here.

Morton sequencing is used to assign consecutive numbers, starting with 0, to the grids in a two-dimensional image as shown in Figure 1. The assignment proceeds in a NW, NE, SW, and SE directional order such that the correspondence between the position of a pixel on an image and its Morton number is a one-to-one mapping. All linear quadtree constructions (Gahegan, 1989; Gargantini, 1982a; Gargantini, 1982b; Lauzon *et al.*, 1985; Rosenfeld *et al.*, 1984; Wang, 1991) are developed based on this idea except that Abel (1985) assigned the number sequence in varied directions. This unique mapping relationship between the position and the Morton number enables the proposed 2DTE technique to handle correctly the positions for regional data in an image. In addition to Morton sequencing, the required number of decompositions for an image, i.e., the level of recursion for subdivision, is another important factor in the development of the 2DTE linear quadtree construction algorithm. We need it to reconstruct the image using quaternary codes. The number of decompositions delimits the range of the Morton sequence for the currently processing quadrant in the decoding routine. In contrast, the coding technique used by Gargantini (1982a) has the number of decompositions embedded in quaternary codes. For the proposed 2DTE method, the number of decompositions to construct a linear quadtree is directly kept as the first record in the quaternary code file. We have a record of four at the front of the linear array if the sample image in Figure 2 is processed.

The concept of a template matching operation helps us to decide whether a further decomposition is necessary in the construction of 2DTE linear quadtree codes. The decision for a quaternary region to be white, black, or in mixed form is considered a matching process between a template and a test image. We may conceive that there exist some templates

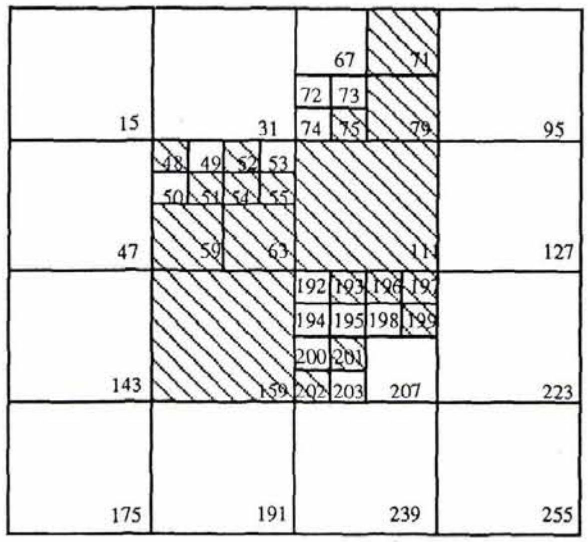


Figure 2. A sample image with Morton numbers.

of various sizes that can be used to determine the decisions. However, we actually include no template in the proposed coding processing; instead, we extract this concept to design a coding technique based on a last homogeneous pixel in the proposed coding algorithm. So, the three templates shown in Figure 3 are conceptually used in the proposed 2DTE method. Digit "0" is set for regions containing only white pixels; digit "1" is set for regions containing only black pixels; but, digit "2" is set for regions containing both white and black pixels. The region of code containing digit "2" implies that further decomposition is done to divide that grid into four quadrants.

Encoding

Once Morton numbers have been assigned to all 2^n by 2^n pixels in the original image, the first decomposition begins to partition that image into four subimages. These four quadrants are encoded in NW, NE, SW, and SE sequences. The selection of digits for each quadrant is based on pixels in that region. We apply one digit for each quadrant, appended from left to right. So, a fixed quaternary code of four digits is used to represent the status of each decomposition. The encoding process continues until all quaternary codes contain no more "2s" in a decomposition routine; i.e., all partitioned regions on an image contain only uniform intensity. To illustrate the proposed encoding process, the image shown in Figure 2 was tested again according to the proposed idea. The results are shown in Figure 4. Figure 4a depicts the connection among all linear quaternary codes for all regional

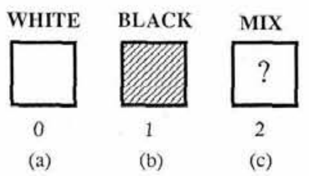


Figure 3. (a) If a leaf is WHITE, set digit 0. (b) If a leaf is BLACK, set digit 1. (c) Digit 2 is set for a mixed area.

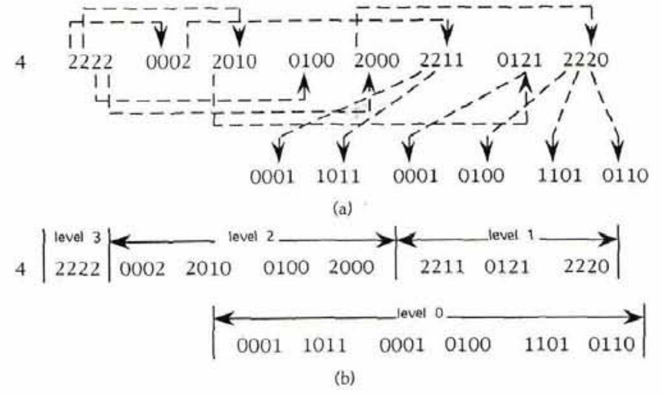


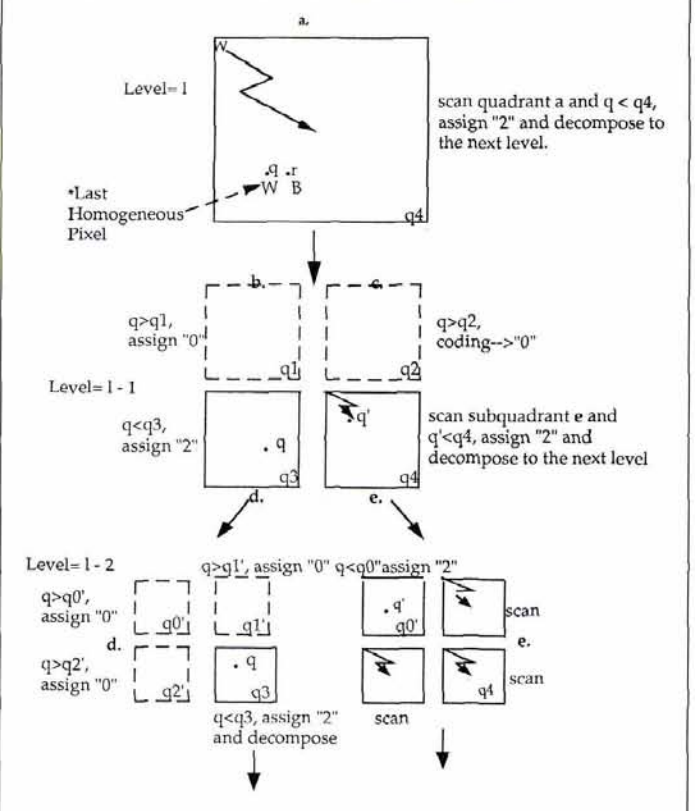
Figure 4. The quaternary codes for the image in Figure 2 using the 2DTE technique.

data of uniform intensity; all directional edges link the recurrence of decomposition. Figure 4b shows the resulting linearly coded words. The first record has a value of 4 to indicate that there are four decompositions for this image; the first record is also named as the level number. The latter is counted in decreasing order $L-1$, $L-2$, ..., 1; i.e., the first partition has level number L and the last partition has level number 1. The second record has code 2222, implying that the first decomposition is not enough; it has to be partitioned into four quadrants 0002, 2010, 0100, and 2000, respectively. Again, three quadrants among them at this level have to be further partitioned. The decomposition stops at the fourth time when all quaternary codes lack any digit "2." According to this example, we find that the number of digits "2" in the codes of the upper level implies the number of quaternary codes in the next decomposition routine. In addition, the first record of digit "4" represents the level number L of the encoding routine. This level number L has the following relation to the Morton number for each region on an image:

$$4^L = 1 + \text{the last Morton number of a region.}$$

L equals 4 in this example. The purpose of the level number is retained to compute the region size for the coding algorithm. At the time of execution, the last Morton number of a specific region has to be retained to address the scope of the next decomposition. Therefore, the NW quadrant for the first decomposition has a Morton sequence from 0 to 63; the NW quadrant for the next decomposition has Morton numbers from 0 to 15, etc.

The proposed 2DTE encoding routine recursively scans all pixels on the image. The scanning is in a hierarchical sequence, i.e., the $L - 1$ level (decomposition) cannot be started until the upper level L is completed. Thus, each pixel may be repetitively visited n times to determine a uniform intensity if the image is of 2^n by 2^n resolution. The determination would require $O(n \times 2^{2n})$ time complexity to complete the whole process. It is naturally not an efficient coding algorithm to satisfy the criterion of a linear computational time complexity. We have therefore necessarily to develop another treatment to apply the concept of template matching to determine the requirement of a decomposition. Once the encoding routine begins to visit from the northwestern corner on the image, it remembers the intensity of the first pixel. The scanning continues until a pixel of intensity different from the first one is reached. We record the Morton number of this specific pixel, called the last homogeneous pixel (LHP). We have the following definition of the LHP:



a.
Level= l
scan quadrant a and q < q4, assign "2" and decompose to the next level.
q r
W B
*Last Homogeneous Pixel
q4
b.
c.
q>q1, assign "0"
q>q2, coding-->"0"
q1
q2
Level= l - 1
q<q3, assign "2"
q
q3
q'
scan subquadrant e and q'<q4, assign "2" and decompose to the next level
q4
d.
e.
Level= l - 2
q>q1', assign "0" q<q0"assign "2"
q>q0', assign "0"
q0'
q1'
q'
q0'
scan
d.
e.
q>q2', assign "0"
q2'
q
q3
q4
scan
q<q3, assign "2" and decompose
scan

Figure 5. Usage of the LHP and the scanning sequence.

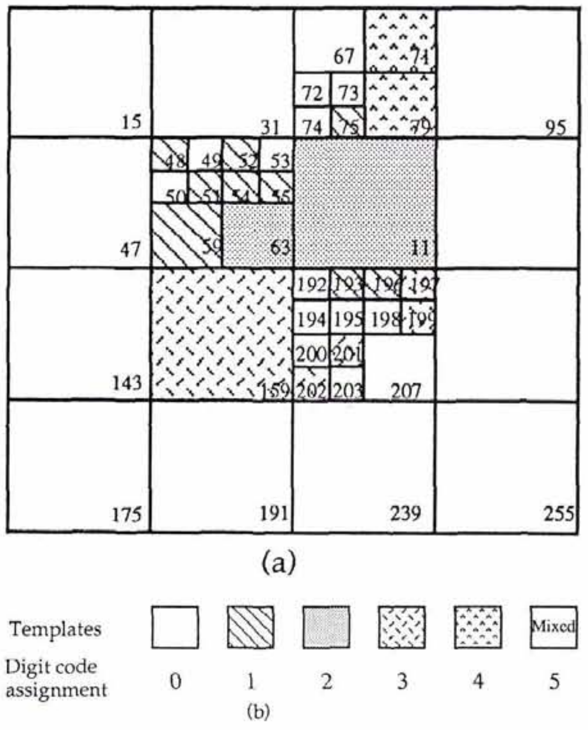


Figure 6. (a) The multicolored image with Morton numbers. (b) The templates and digit code assignment for the 2DTE technique.

In any currently processing quadrant, an LHP is a particular pixel that is the first in which the intensity differs from all previously accessed pixels.

With the usage of the LHP, a decision on further decomposition of the currently processed quadrant is made if the Morton number of the LHP is less than or equal to the Morton number of the last pixel of that quadrant. A linear code "2" is assigned to that quadrant and further decomposition for that quadrant is determined. So, as the encoding routine begins to scan a quadrant, it records the intensity of the first pixel initially at the northwestern corner of that quadrant. Thereafter, the encoding routine visits the quadrant in Morton sequence until it finds the LHP. The Morton number of that LHP is then recorded. With the help of the Morton number of the LHP, we eliminate the necessity of repetitively visiting all previously accessed pixels. The elimination is derived because there are particular relationships among the Morton numbers of the LHP and four subquadrants.

When a decomposition is decided, the four last Morton numbers of its four subquadrants can be computed first. The relationship between the Morton number of the LHP and the four last Morton numbers of the four subquadrants provides the entry point of the next scanning. The entry point of the next scanning fits one of the following cases:

- If the Morton number of the LHP (for the upper level) is greater than the last Morton number of a subquadrant, the subquadrant is already accessed; it must have an intensity of either black or white. A linear code of either a "0" or a "1" can be assigned to that subquadrant immediately.
- In contrast, if the Morton number of the LHP is less than the last Morton number of a specific subquadrant, this subquadrant has both white and black pixels; the linear code "2" is assigned and further decomposition is undertaken.

In Figure 5, we give a clear illustration of how the proposed LHP works. Suppose a quadrant at level l has background pixels of white intensity up to the pixel of Morton number q. The pixel of Morton number r is the first pixel for an object of black intensity. As the pixel r is selected as the LHP that indicates that this quadrant contains both background and object pixels, so the decision of further decomposition of this quadrant is made and a digit "2" is assigned to this quadrant. When the decomposition is done, two subquadrants $q1$ and $q2$ at level $l-1$ are assigned to have digit "0" immediately without revisiting because the last Morton numbers of these two subquadrants are less than that of LHP q. Subquadrant $q3$ is assigned a digit "2" because the Morton number of the LHP q is less than the last Morton number of $q3$. The scan for the level $l-1$ begins from the first pixel of subquadrant $q4$. Subquadrants $q0'$, $q1'$, and $q2'$ at level $l-2$ are assigned digit "0" directly without revisiting; $q3'$ and $q0''$ are assigned digit "2" directly; scanning at the level $l-2$ is required for only the other three subquadrants. Hence, the necessity to revisit pixels accessed before is eliminated. The encoding algorithm is described in Appendix A.

Decoding

The decoding routine first reads the level number in order to determine the whole image size. It computes the scope of a Morton sequence for each quadrant and determines the last Morton number for each region. Afterward, the code sequence is read. The decoding routine is processed repeatedly until all codes are encountered. Any digit "2" in codes of the upper level implies a region with uniform intensity not yet met. The color of a region is determined when the codes of only "0" or "1" are reached. The decoding algorithm is listed in Appendix B.

Table 1. Comparison of the Results for Four File Structures for the Image in Figure 2

Record	The proposed scheme 2DTE codes	Gargantini* encoding Linear codes	2-dimension run-encoding		one record per leaf		
			Morton#	Attr.	Morton#	level	Attr.
0	4	0300	47	W	15	2	W
1	2222	0303	48	B	31	2	W
2	0002	0310	50	W	47	2	W
3	2010	0312	52	B	48	0	B
4	0100	0313	53	W	49	0	W
5	2000	032X	63	B	50	0	W
6	2211	033X	67	W	51	0	B
7	0121	101X	71	B	52	0	B
8	2220	1023	74	W	53	0	W
9	1001	103X	79	B	54	0	B
10	1011	12XX	95	W	55	0	B
11	0001	21XX	111	B	59	1	B
12	0100	3001	143	W	63	1	B
13	1101	3010	159	B	67	1	W
14	0110	3011	192	W	71	1	B
15		3013	193	B	72	0	W
16		3021	195	W	73	0	W
17		3022	197	B	74	0	W
18			198	W	75	0	B
19			199	B	79	1	B
20			200	W	95	2	W
21			202	B	111	2	B
22			255	W	127	2	W
23					143	2	W
24					159	2	B
25					175	2	W
26					191	2	W
27					192	0	W
28					193	0	B
29					194	0	W
30					195	0	W
31					196	0	B
32					197	0	B
33					198	0	W
34					199	0	B
35					200	0	W
36					201	0	B
37					202	0	B
38					203	0	W
39					207	1	W
40					223	2	W
41					239	2	W
42					255	2	W

*Each digit needs three bits for representation of 0, 1, 2, 3, and the mark X

Using the 2DTE Scheme for Multicolor Problems

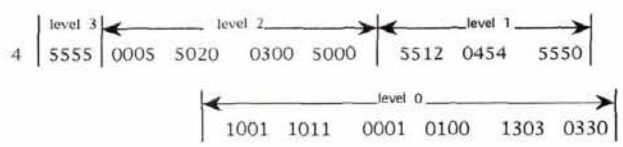
The proposed 2DTE coding scheme is also workable for handling problems of multicolor images. While a thematic map, being an example, may contain more than one object, the proposed 2DTE scheme can be easily tailored to construct the corresponding quaternary codes. Conceptually, the three templates used for a black-and-white (B/W) image are extended to be 2^m templates with which 2^{m-1} objects are encoded. Each object is represented by m bits, and a quaternary code consists of $4m$ bits. A multicolor image is handled using the same coding process as that of a B/W image. So, a linear list of quaternary codes, as shown in Figure 7, is derived if the sample image shown in Figure 6 contains four objects. The illustration shown in Figure 7 clearly demonstrates the capability of the proposed 2DTE scheme for multicolor images.

Empirical Results and Performance Analysis

The performance of the proposed 2DTE technique is analyzed and compared in terms of memory complexity and temporal complexity. The performance is compared among the proposed 2DTE technique, Gargantini's (1982a) coding, the 2DRE coding of Lauzon *et al.* (1985), and a general complete linear quadtree (one record for each leaf) technique to reveal the superiority of the proposed 2DTE technique. Empirical results validate the feasibility of the proposed 2DTE technique. The potential of 2DTE scheme for multicolor problems is also explained.

Empirical Results

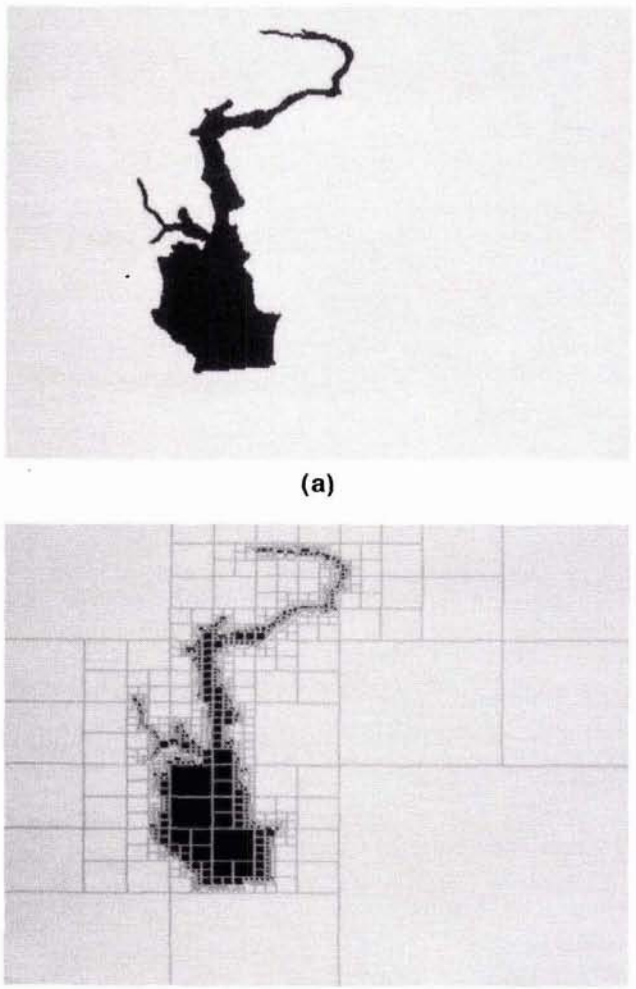
In order to test the characteristics of the proposed 2DTE technique, we used the 2^4 by 2^4 image data in Figure 2 to compare the results of the four methods; all results are listed in Table 1. In the first column of Table 1, there are 14 quater-



```
4  5555  0005 5020 0300 5000  5512 0454 5550
   1001 1011 0001 0100 1303 0330
```

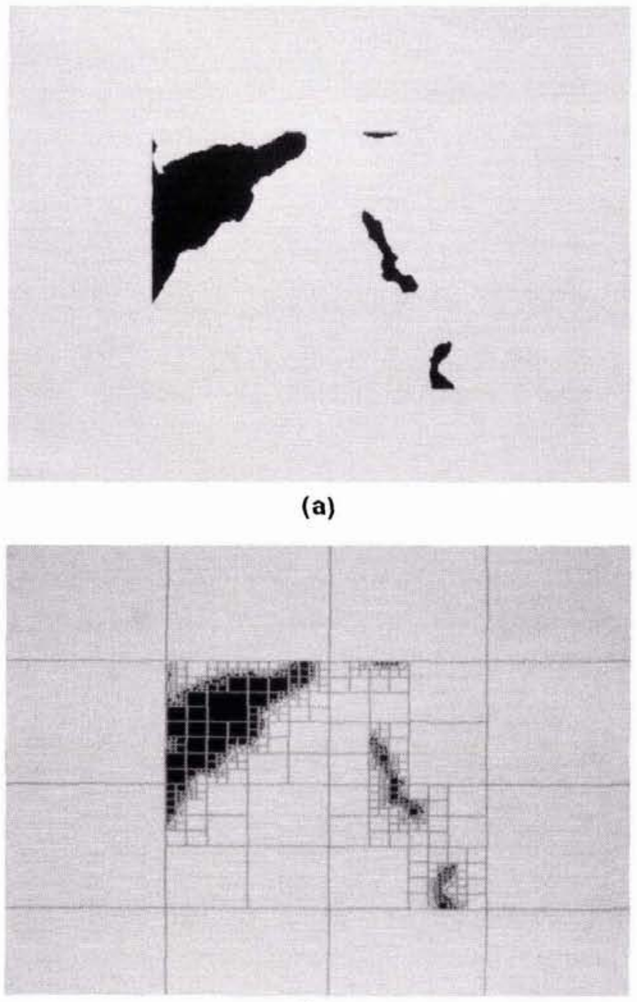
Figure 7. The resulted linear quaternary codes.

nary codes for the proposed 2DTE technique; the first digit, "4," is the number of the level. Each quaternary code needs exactly eight bits, of which two bits are used for each digit; so 112 bits in total are required. The second column is for Gargantini's (1982a) coding. There are 18 records; each record needs 11 bits in presentation because each digit requires three bits for possible codes 0, 1, 2, 3, and the marker X, but no node can start with an X. Hence, the required memory is 198 bits. The results for the 2DRE coding by Lauzon *et al.* (1985) are listed in the third column. 2DRE encoding needs nine bits for each record, of which eight bits are required for the Morton sequence and the other one bit is for an attribute. There are 23 records that induce a memory size 207 bits. The fourth column lists the result of a general complete linear quadtree. There are 42 records; each record needs 11 bits, of which eight bits are used for the Morton sequence, two bits are for the level number, and the final bit is for the attribute. A memory size of 462 bits in total is used. The proposed 2DTE technique clearly requires the least memory space. To test further the characteristics of the proposed 2DTE technique, we completed two tests to compare the results. In the first test, we used the generally applied floodplain map embedded in a 512 by 512 grid shown in Figure 8a; the second image of resolution 256 by 256 shown in Figure 9a is the agricultural region of the Taipei area. The corresponding processed images are shown in Figures 8b and 9b, which illustrate the lossless encoding results. All statistical data for comparison are collected in Tables 2 and 3. The comparisons are derived with four methods. The proposed 2DTE technique clearly requires the least memory space. The results listed in Tables 2 and 3 clearly demonstrate the advantage of the proposed 2DTE technique.


(a)

(b)

Figure 8. (a) The original floodplain image of 512 by 512 pixels. (b) The decomposed result of the floodplain image.


(a)

(b)

Figure 9. (a) The original Taipei agricultural image of 256 by 256 pixels. (b) The decomposed result of the Taipei agricultural image.

Performance Analysis

The theoretical comparison of the proposed 2DTE technique with other methods is analyzed based on an image size of 2^n by 2^n in which n is the resolution parameter. The required amount of memory space is approximately computed by the

TABLE 2. STORAGE SPACE COMPARISON

Image size: 512 by 512. Image name: Florida flood area				
File structure	complete	Gargantini	2DRE	2DTE
Nodes	3313	1544	1511	1104
Bits	62947	40144	28709	8832

TABLE 3. STORAGE SPACE COMPARISON

Image size: 256 by 256. Image name: Taipei farm area				
File structure	complete	Gargantini	2DRE	2DTE
Nodes	1846	866	771	615
Bits	31382	19918	13107	4920

product of the number of regions and the length of a code. Each quaternary code of the 2DTE technique requires a constant eight bits; Gargantini's code needs $3(n-1)+2$ bits; the 2DRE approach of Lauzon *et al.* needs $2n+1$ bits to encode a key; while $3n+1$ bits are required for the complete coding method. The total required memory is compared by considering the worst case. Figure 10 shows two cases for comparison. Figure 10a is selected for both the proposed 2DTE and Gargantini's technique, while Figure 10b is the worst case for the 2DRE coding of Lauzon *et al.* A simple computation reveals that Gargantini's coding technique requires

$$(3(n-1)+2) \times (3 \times 4^{n-1}) \approx 9n \times 4^{n-1} \text{ bits}$$

of which $3\times4^{n-1}$ black nodes are recorded. The 2DRE coding method of Lauzon *et al.* needs $(2n+1)\times4^n$ bits because the runlength code is set to each pixel. The proposed 2DTE coding method needs

$$8 \times \sum_{L=1}^{n} 4^{n-1} = 8 \times \frac{4^{n-1}}{3} \approx 8 \times 4^{n-1}$$

bits. For the complete coding method, the memory space required is $(3n+1)\times4^n$. The proposed 2DTE coding technique can obviously have less memory space than the other three coding methods, and the proposed 2DTE method has a requirement $1/n$ to Gargantini's method and the 2DRE coding techniques of Lauzon *et al.*

Temporal complexity is another factor for comparison. The temporal complexity, $O(n)$, of the 2DTE method is linear with the image size, because the image is accessed only once during encoding and decoding. The required execution duration is exactly the same as for other methods such as those of Gargantini or Lauzon *et al.*

Conclusion and Future Research

We proposed the 2DTE coding technique for the linear quadtree representation, based on the concept of template matching and Morton sequencing. We suggest a fixed length of eight bits to encode a quaternary key. With the appropriate data structure, we have a linear quadtree coding process that reduces memory complexity without adding extra computational complexity. The proposed 2DTE technique is seen to have improved performance over all previous research. Experimental results verify the applicability of the proposed 2DTE technique for linear

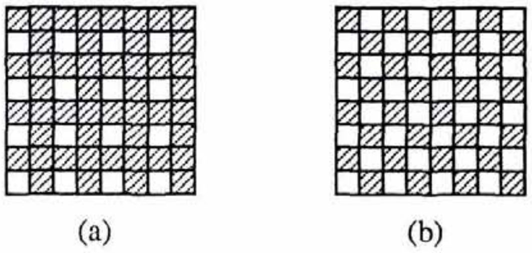


Figure 10. (a) The worst case for Gargantini's coding and the proposed 2DTE coding. (b) The worst case for 2DRE coding.

quadtree construction. It is shown to provide improvements in the memory requirement for geometric data processing. Future research can focus on the following:

- Extending the proposed 2DTE technique for a multicolored image; the necessity of generating multiple quadtrees for each object can then be completed in one encoding routine;
- Developing basic operators for geographical data processing, intersection, union, etc., for the encoded keys of the 2DTE technique in a GIS; and
- Extending the proposed 2DTE coding technique to the octree structure applicable to areas of computer vision or medical images.

Acknowledgment

This work was supported by the National Science Council, R.O.C., under Grant NSC 82-0408-E-123-003.

References

Abel, D.J., 1984. A B+-tree structure for large quadtrees, *Comput. Vision, Graphics Image Process*, 27:19–31.

———, 1985. Some elemental operations on linear quadtrees for geographic information systems, *The Computer Journal*, 28:73–77.

Dyer, C.R., 1982. The space efficiency of quadtrees, *Comput. Graphics Image Process*, 19:335–348.

Dyer, C.R., A. Rosenfeld, and H. Samet, 1980. Region representation: Boundary codes from quadtrees, *Comm. ACM*, 23:171–179.

Gahegan, M.N., 1989. An efficient use of quadtrees in a geographic information system, *Int. J. Geograph. Inf. Syst.*, 3:201–214.

Gargantini, I., 1982a. An effective way to represent quadtrees, *Comm. ACM*, 25:905–910.

———, 1982b. Translation, rotation and superposition of linear quadtree, *Int. J. Man-Machine Studies*, 18:253–263.

Hunter, G.M., and K. Steiglitz, 1979. Operations on images using quadtrees, *IEEE Trans. Pattern Anal. Mach. Intell*, PAMI-1:145–153.

Klinger, A., and C.R. Dyer, 1976. Experiments on picture representation using regular decomposition, *Comput. Graphics Image Process*, 5: 68–105.

Klinger, A., and M.L. Rhodes, 1979. Organization and access of image data by areas, *IEEE Trans. Pattern Anal. Mach. Intell.*, PAMI-1:50–60.

Lauzon, J.P., D.M. Mark, L. Kikuchi, and J.A. Guevara, 1985. Two dimensional run encoding for quadtree representation, *Comput. Vision, Graph. & Image Process*, 30:56–69.

Morton, G.M., 1996. *A Computer Oriented Geodetic Data Base, and a New Technique in File Sequencing*, IBM Canada Ltd., 1 March.

Rosenfeld, A., H. Samet, C. Shaffer, and R.E. Webber, 1984. Use of hierarchical data structures in geographical information systems, *Proceedings, Int. Symp. Spatial Data Handling*, Zurich, Switzerland, pp. 392–411.

Samet, H., 1980. Region representation: Quadtrees from boundary codes, *Comm. ACM*, 23:163–170.

———, 1981. An algorithm for converting rasters to quadtrees, *IEEE Trans. Pattern Anal. Mach. Intell*, PAMI-3:93–95.

———, 1984. The quadtree and related hierarchical data structures, *ACM Comput. Surveys*, 16:187–260.

———, 1989. *The Design and Analysis of Spatial Data Structure*, Addison-Wesley Publishing Co., Inc. New York.

Shaffer, C.A., and H. Samet, 1987. Optimal quadtree construction algorithms, *Comput. Vision, Graph. & Image Process*, 37:402–419.

Tanimoto, S., and T. Pavlidis, 1975. A hierarchical data structure for picture processing, *Comput. Graphics Image Process*, 4:104–119.

Wang, F., 1991. Relational-linear quadtree approach for two-dimensional spatial representation and manipulation, *IEEE Trans. Knowledge and Data Engineering*, 3:118–122.

Webber, R.E., and M.B. Dillencourt, 1989. Compressing quadtrees via common subtree merging, *Pattern Recognition Letters*, 9:193–200.

(Received 8 December 1994; accepted 18 January 1995; revised 13 March 1995)

Appendix A. Encoding Algorithm

```
ENCODING( )
Begin
    LAST_MNUM<--- IMAGE_WIDTH * IMAGE_HEIGHT-1;
    LEVEL_NUM <--- LEVEL(LAST_MNUM) ;
    c <--- 0 ;   r <--- 0 ;   i <--- 0 ;
    NUM_of_2 <--- 0 ;
    2DTE[ c] <--- LEVEL_NUM;
    c <--- c + 1 ;
    LEVEL_NUM <--- LEVEL_NUM - 1 ;
    for ( k <--- 0 to 3 )
    begin
        QRANT_NUM[ k ] <--- DIVIDE_QRANT(LEVEL_NUM, LAST_MNUM, k ) ;
        PIXELa[ i ].num <--- VISIT_QRANT(QRANT_NUM[ k ] , LEVEL_NUM) ;
        PIXELa[ i ].color<--- COLOR(QRANT_NUM[ k ] , LEVEL_NUM) ;
        if (PIXELa[ i ].num < QRANT_NUM[ k ])
        begin
            2DTE[ c] <--- 2 ;
            MORTONa[ r ] <--- QRANT_NUM[ k ] ;
            NUM_of_2 <--- NUM_of_2 + 1 ;
            r <--- r + 1 ;  i <--- i + 1 ;
         end ;
         else
            2DTE[ c ] <--- PIXELa[ i ].color ;
         c <--- c + 1;
    end ;
    r <--- 0 ;  s <--- 0 ;  i <--- 0 ;  j <--- 0 ;
    LEVEL_NUM <--- LEVEL_NUM - 1 ;
    while (NUM_of_2 < > 0)
    begin
         NEXTLEVEL_NUM_of_2 <--- 0 ;
        for ( q <--- 1 to NUM_of_2) /*compare LH pixel with quadrant */
        begin
  for ( k <--- 0 to 3)
     QRANT_NUM[ k ] <--- DIVIDE_QRANT(LEVEL_NUM, MORTON[ r ], k ) ;
  k <--- 0 ;   e <--- 0 ;   r <--- r + 1;
  if (PIXELa[ i ].num < QRANT_NUM[ k ] )
  begin
      2DTE[ c ] <--- 2 ;
      PIXELb[ j ].num <--- PIXELa[ i ].num ;
      PIXELb[ j ].color<--- PIXELa[ i ].color ;
      MORTONb[ s ] <--- QRANT_NUM[ k ] ;
      NEXTLEVEL_NUM_of_2 <--- NEXTLEVEL_NUM_of_2 + 1 ;
      c <--- c + 1 ;   i <--- i + 1 ;   j <--- j + 1 ;
      e <--- e + 1 ;   s <--- s + 1 ;
   end ;
   else
   begin
       while(PIXELa[ i ].num > = QRANT_NUM[ k ] )
        begin
            2DTE[ c ] <--- PIXELa[ i ].color ;
            c <--- c + 1 ;   k <--- k + 1 ;   e <--- e + 1 ;
        end ;
        if ( PIXELa[ i ] .num > QRANT_NUM[ k -1] )
        begin
            2DTE[ c ] <--- 2 ;
            PIXELb[ j ].num <--- PIXELa[ i ].num ;
            PIXELb[ j ].color<--- PIXELa[ i ].color ;
            MORTONb[ s ] <--- QRANT_NUM[ k ] ;
            c <--- c + 1 ;   j <--- j + 1 ;
            e <--- e + 1 ;   s <--- s + 1 ;
         end ;
         i <--- i + 1 ;
    end ;
        if (e = < 3)
        begin
            for(k <--- e to 3)
            begin
                PIXELb[ j ].num <---
                VISIT_QRANT(QRANT_NUM[ k ] , LEVEL_NUM) ;
                PIXELb[ j ].color<--- COLOR(QRANTNUM[ k ] , LEVEL_NUM) ;
                if ( PIXELb[ j ].num < QRANT_NUM[ k ] )
                begin
                    2DTE[ c ] <--- 2 ;
                    MORTONb[ s ] <--- QRANT_NUM[ k ] ;
                    s <--- s + 1 ;   j <--- j + 1 ;
                    NEXTLEVEL_NUM_of_2 <---NEXTLEVEL_NUM_of_2 + 1 ;
                 end ;
                 else
                    2DTE[ c ] <--- PIXELb[ j ].color ;
                 c <--- c + 1 ;
             end ;
         end ;
      end ;/*end of for q loop*/
      NUMBER_of_2 <--- NEXTLEVEL_NUM_of_2 ;
      REPLACE (MORTONa[ ] , MORTONb[ ] ) ;
      REPLACE (PIXELa[ ] , PIXELb[ ] ) ;
      LEVEL_NUM <--- LEVEL_NUM - 1 ;
   end ;/*end of while NUMBER_of_2 loop*/
end;
```

Appendix B. Decoding Algorithm

```
DECODING ( )
begin
      c<---0;
      LAST_MNUM <--- $4^{2DTE[c]}$ - 1;
      LEVEL_NUM<--- 2DTE[ c ] - 1 ;
      i <--- 0;
      c <--- c + 1 ;

     NUMBER_OF_2 <--- 0 ;
     for ( k <--- 0 to 3 )
     begin
        QRANT_NUM [ k ] <---
        DIVIDE_QRANT ( LEVEL_NUM , LAST_MNUM , k ) ;
        if ( 2DTE[c] = 2 ) then
        begin
             MORTONa [ i ] <--- QRANT_NUM [ k ] ;
             i <--- i + 1 ;
             NUMBER_OF_2 <--- NUMBER_OF_2 + 1 ;
         end ;
         else
             PRINT_IMG ( 2DTE[ c ], LEVEL_NUM , QRANT_NUM[ k ] ) ;
         c <--- c + 1;
     end ;

          LEVEL_NUM <--- LEVEL_NUM - 1 ;
          i <--- 0 ;
          j <--- 0 ;

      while ( NUMBER_OF_2 < > 0 )
      begin
         NEXTLEVEL_NUM_of_2 <--- 0 ;
       for ( i <--- 1 to NUMBER_OF_2 )
       begin
   for ( k <--- 0 to 3 )
   begin
      QRANT_NUM [ k ] =
      DIVIDE_QRANT ( LEVEL_NUM, MORTONa[ i ], k ) ;
      if ( 2DTE[ c ]  = 2 ) then
      begin
           MORTONb[ j ] <--- QRANT_NUM [ k ] ) ;
           j <--- j + 1 ;
           NEXTLEVEL_NUM_of_2 <--- NEXTLEVEL_NUM_of_2 + 1 ;
      end ;
      else
```

```
        PRINT_IMG ( 2DTE[ c] , LEVEL_NUM, QRANT_NUM [ k ] ) ;
      c <--- c + 1;
    end ;
    i <--- i + 1 ;
  end ;
```

```
            REPLACE(MORTONa[ ] , MORTONb[ ] ) ;
            NUMBER_OF_2 <--- NEXTLEVEL_NUM_of_2 ;
            NEXTLEVEL_NUM_of_2 <--- 0 ;
            LEVEL_NUM <--- LEVEL_NUM - 1 ;
          end ;
        end ;
```

JOIN US IN SUNNY FLORIDA!

ASPRS-RTI
A NEW VISION

1998 ASPRS-RTI Annual Conference
March 30—April 3, 1998
Tampa Convention Center
Tampa, Florida

A New Vision not only describes our industry at a time when new sensors and technolgoies are emerging at a rapid pace, but also captures our enthusiasm as ASPRS unveils its new conference look and feel.

Conference Schedule

Committee Meetings
Monday - Wednesday
March 30 - April 1

Full-Day Workshops
Monday - Tuesday
March 30 - March 31

Educational Sessions and Exhibits
Wednesday - Friday
April 1 - April 3

Conference Hotels

Hyatt Regency Tampa
Wyndham Harbour Island Hotel
Holiday Inn Select

Resource Technology

As a professional involved with photogrammetry, environmental management, remote sensing, geographical sciences, natural resources, mapping, land information systems, or geodesy, you won't want to miss this outstanding opportunity to see products from more than 100 vendors and review the latest issues in your industry. Plan now to attend carefully organized technical sessions targeting your specific needs and interests.

Technical areas include:

- Sensors
- GIS
- Ecological Management
- Natural Resources
- Image Processing
- Potpourri

72 Technical Sessions

Full Registration Fees

Members

$250	$310	$350
Early-Bird*	Advance**	On-Site

Non-Members

$350	$410	$450
Early-Bird*	Advance**	On-Site

Students

$35	$45	$55
Early-Bird*	Advance**	On-Site

* Deadline is January 15, 1998. ** Deadline is March 1, 1998.

Opportunities in Education

Interact with Industry Colleagues

Production Information

Preliminary Programs will automatically be mailed to ASPRS and RTI members in November. If you would like to be added to the mailing list, contact ASPRS at 301-493-0290 x20 or e-mail us at meetings@asprs.org.

For up-to-date information, please be sure to check out the 1998 ASPRS-RTI Annual Meeting web site sponsored by the Florida Host Committee. This page contains all you need to know about the meeting.

http://big.stpt.usf.edu/~tampa