TELKOMNIKA, Vol. 13, No. 4, December 2015, pp. 1319~1329
ISSN: 1693-6930, accredited A by DIKTI, Decree No: 58/DIKTI/Kep/2013
DOI: 10.12928/TELKOMNIKA.v13i4.1899

A Neighbor-finding Algorithm Involving the Application of SNAM in Binary-Image Representation

Jie He[1], Hui Guo[1], Defa Hu*[2]
[1]School of Information and Electronic Engineering, Wuzhou University, Wuzhou 543002, Guangxi, China
[2]School of Computer and Information Engineering, Hunan University of Commerce, Changsha 410205, Hunan, China
*Corresponding author, e-mail: hdf666@163.com

Abstract

In view of the low execution efficiency and poor practicability of the existing neighbor-finding method, a fast neighbor-finding algorithm is put forward on the basis of Square Non-symmetry and Anti-packing Model (SNAM) for binary-image. First of all, the improved minor-diagonal scanning way is applied to strengthen SNAM's adaptability to various textures, thus reducing the total number of nodes after coding; then the storage structures for its sub-patterns are standardized and a grid array is used to recover the spatial-position relationships among sub-patterns, so as to further reduce the complexity of the neighbor-finding algorithm. Experimental result shows that this method's execution efficiency is significantly higher than that of the classic Linear Quad Tree (LQT)-based neighbor-finding method.

Keywords*: Neighbor-Finding; SNAM for Binary-Image; Minor-Diagonal Scanning Mode; Grid Array*

1. Introduction

As the basis of a variety of image operations such as the calculation of images' geometric properties, the labeling of connected domains, images' edge detection, etc. [1], neighbor-finding algorithm can directly affect the execution efficiency of these operations. The expressive methods adopted in digital image and their adjacent definitions decide the performance and space-time complexity of neighbor searching algorithm. The original neighbor-finding algorithm is executed in the two-dimensional array of image, however, due to the large storage space the two-dimensional array occupies and the wide search scope involved in the pixel-pixel traversal, the neighbor-finding algorithm based on pixel array has an unsatisfactory efficiency. The widely used Methods such as quadtree [2], linear quadtree [3] can reduce basic image representation units, save the storage space and enable the rapid "block-block" execution of image operation. Literature [4-6] separately puts forward neighbor-finding algorithms based on the quadtree and linear quadtree representations, which have significantly improved the efficiency of some image processing operations [7-9]. However, because the neighbor relationship is not a one-to-one relationship, this kind of method cannot be easily performed in the actual image operation. Therefore, as new and more simplified image expressive methods emerge continuously, people are still seeking for higher-efficient neighbor searching methods.

The method of non-symmetry and anti-packing image representation achieves optimal asymmetric image segmentation so as to obtain a higher efficiency of representation than that of the quadtree method [10], and derives a series of high-efficient image processing algorithms [11-14]. Literature [15] proposes a binary image representation method which uses square, isolated point, line segment as the basic representation units in the NAM method. With regular shape and simple structure, these basic representation units can serve as excellent basis for further developing other image processing algorithms. However, there remain some deficiencies with this method, restricting its supportive ability for other image processing procedures: Its raster scanning means is prone to producing more sub-schemes when performing reverse arrangement in images with rich massive textures; the separate standalone storage structure of the three sub-schemes also increases the algorithm's space-time complexity. Besides, although the coordinates of the sub-schemes have been recorded, there is a lack of description about the

Received August 26, 2015; Revised October 25, 2015; Accepted November 13, 2015

relation of spatial position among them, which typically entails traversal over all storage queues in the derivative algorithm.

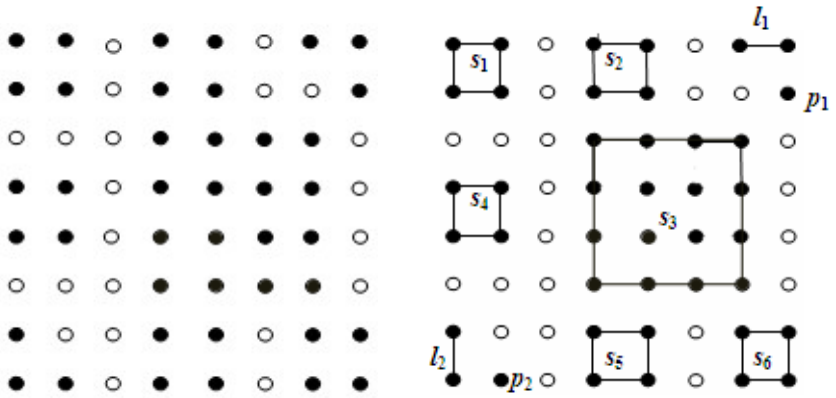
Since the efficiency of neighbor searching algorithm depends directly on the relative positional relation between and quantity of basic representative units, solutions to the above problems are the key to high-efficient neighbor searching on SNAM. The author begins by utilizing the structural characteristic of square to include sub-scanning along the back-diagonal direction in raster scanning and enhance the adaptability of reverse arrangement to massive textures; and then uses a same data structure to record three different sub-schemes so that the storage and operating mode can be uniformed across different sub-schemes; next the author constructs grid matrix, an intermediate structure, to restore the positional relation among the sub-schemes and reduce the searching range for neighbor. Through the above measures, the neighbor searching is implemented on the result of SNAM coding, and compared experimentally with the LQT-based neighbor searching method to demonstrate the efficacy of this work.

2. SNAM for Binary-Image and Its Optimization

2.1. Square NAM-Based Representation Method of Binary Image

The image expressive method of transform domain usually has higher compression ratio than methods of spatial domain [16], but the latter can reserve the image's original local textural characteristics and positional relation more effectively. SNAM for binary-image is a nondestructive representation method, which is based on the space domain and uses three types of sub-patterns - square, line segment and isolated point - as the basic image representation unit. Its process is as follows: square, line segment and isolated points at different scales are extracted in a raster scan mode from a packed binary image through the anti-packing algorithm, and corresponding parameters of these sub-patterns are recorded and stored, thus forming the representation of the given binary image.

Figure 1 shows the result of anti-packing, coding and storing a binary image with SNAM. Figure 1(a) is the given original binary image T with a size of $2^n \times 2^n$ (n =3). Figure 1(b) shows the result of SNAM anti-packing in the raster scan mode. Six square sub-patterns (s_1, s_2, s_3, s_4, s_5 and s_6), two line segments (l_1 and l_2) and two isolated points (p_1 and p_2) are extracted from the original image. Figure 1(c) is the representation of T by using the above-mentioned sub-patterns.



(a) An 8×8 binary image T (b) The SNAM anti-packing (c) The result of of T

Figure 1. The process of anti-packing a binary image with SNAM

According to SNAM, the items need to be recorded are: for square, the coordinates *sp* of the upper left vertex (i.e., the starting point) and the side length *side*; for line segment, the coordinates p_1 of the starting point and the coordinates p_2 of the end point; for isolated point, only the coordinates of the vertex. The storage structures for all sub-patterns are as shown in Figure 2.

sp	*side*

(a) square

p_1	p_2

(b) line

i_ p

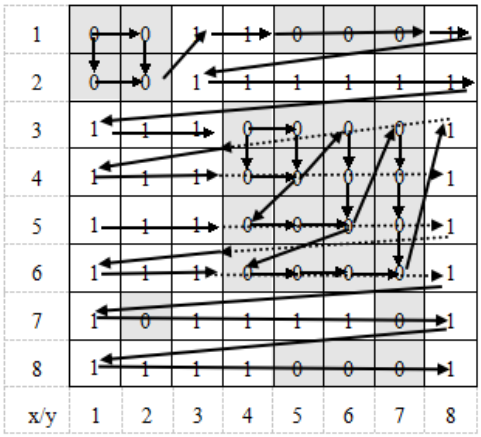
(c) isolated point

Figure 2. The process of anti-packing a binary image with SNAM

SNAM can also bring some problems for image processing. Literature [17] notes that the raster scan mode has a good performance toward images with unobvious blocky features, but shows low efficiency toward images with obvious blocky features. In addition, the diversity of SNAM's storage structures for three types of sub-patterns and its original anti-packing algorithm lead to the emergence of both horizontal and vertical line segments in the coding result, further increasing the complexity of subsequent image processing.

2.2. Optimization of the Scanning Mode

Raster scan is characterized by the line-by-line horizontal scan, and a combination of raster scan and block scan will produce a stronger adaptability to images with blocky texture. The horizontal direction is still the main scanning direction, but when unmarked black point is detected, the structural characteristics of square are used and determining the pixel values of pixels on the minor-diagonal is regarded as the core algorithm - thus the minor-diagonal scanning mode is formed. Its principle is shown in Figure 3. Therein, Figure 3(a) is the diagram of the minor-diagonal scanning direction, where the dotted line indicates that after these points form a square, they needn't go through horizontal scan any more. Figure 3(b) shows the serial numbers of all pixels in the minor-diagonal scanning process. This mode not only has a stronger adaptability, but only lowers the complexity of SNAM by only producing horizontal line segments.



(a) Direction for minor-diagonal scan

1	1	2	4	5	6	7	8	9
2	2	3	10	11	12	13	14	15
3	16	17	18	19	20	22	25	29
4	30	31	32	20	21	23	26	33
5	34	35	36	22	23	24	27	37
6	38	39	40	25	26	27	28	41
7	42	43	44	45	46	47	48	49
8	50	51	52	53	54	55	56	57
x/y	1	2	3	4	5	6	7	8

(b) Pixel sequence in minor-diagonal scan

Figure 3. Principle of minor-diagonal scan

The specific procedure of the SNAM algorithm based on minor-diagonal scan (Minor Diagonal-SNAM, or MD-SNAM) is as follows:

Input: a binary image *T* with a size of *n*×*n.*

Output: *V_s*—the set of square sub-patterns, *V_l*—the set of line segment sub-patterns, *V_p*—the set of isolated point sub-patterns.

Step 1 Search for the unmarked point with zero pixel value by starting from the pixel first seen when accessing *T* in this circle, and use the found pixel as the upper vertex of the square sub-pattern, then record its coordinates (*sp_x*, *sp_y*), and initialize the side length of the square sub-pattern with *side*.

Step 2 Search for unmarked pixels on the back diagonal of the square with *side*-long sides, and judge whether their pixel values are 0: if they are not 0, switch to Step 1.

Step 3 Make *side*=*side*+1, and continue to perform Step 2 until no more square sub-pattern forms; store the found square in *V_s*, and mark all the pixels covered by the square with *side*-long sides.

Step 4 Rotate Steps 1~3 until no new square sub-patterns can be extracted out.

Step 5 Search for the unmarked point with zero pixel value by starting from the pixel first seen when accessing *T* in this circle, and record its coordinates (l_x_1, l_y_1).

Step 6 Judge whether the value of the neighboring pixel at the horizontal right side of the point (l_x_1, l_y_2) is zero: if it is, continue the search toward the horizontal right side until the line segment stops enlarging, then record the line segment's right endpoint's coordinates (l_x_2, l_y_2), and store the line in *V_l*, and then mark all the pixels that it covers; if it is not zero, go to Step 5.

Step 7 Rotate Step 5 and Step 6 until no new line segment sub-patterns can be extracted out.

Step 8 Search for the unmarked point with zero pixel value by starting from the pixel first seen when accessing *T* in this circle, and record its coordinates (*p_x*, *p_y*); then store it in *V_p* and mark this point.

Step 9 Rotate Step 9 until no new isolated-point sub-pattern can be extracted out.

Step 10 Output *V_s*, *V_l* and *V_p*.

2.3. Optimization of the Storage Structure

SNAM has designed storage structures separately for three different types of basic representation units, which, while can ensure compact storage, means the same SNAM-based image operation algorithm needs to separately process different sub-patterns. Therefore, using the same kind of simple data structure to record different sub-patterns is the key to lowering the complexity of the subsequent image processing algorithm.

For the three types of sub-patterns - square, line segment and isolated point - involved in SNAM, a uniform storage structure can be defined as follows:

x	*y*	*Length*

Figure 4. The uniform storage structure for SNAM sub-patterns

Therein, *x* represents the abscissa of the starting point of the sub-pattern, *y* represents its ordinate, and *Length* represents the sub-pattern's side length. In the coding, the *Length* of square and isolated point is stored as int data, while the *Length* of line segment is stored as *string* data. Thus the sub-patterns can be distinguished by the *Length* value and data type: if the sub-pattern is square, its *Length* must be greater than 1 and belong to int data; if the sub-pattern is isolated point, which can be regarded as a square with a side Length of 1, then its *Length* must be equal to 1 and belong to int data; and if the sub-pattern is line segment, its side length (i.e., its length) *Length* is also greater than 1 but belongs to *sting* data. In this way, a uniform sub-pattern set can be established to realize rapid traversal, thus providing more effective supports for various image operations.

3. Neighbor-Finding Based on MD-SNAM

3.1. Definition of Neighbor

The word "neighbor" is used for depicting the adjacent relation among image areas that image representation units correspond to, and its definition varies depending on specific image representation methods. Literature [18] has presented the definition of neighbor in the quadtree method, and considering that both SNAM and the quadtree representation method express image source as the record set of local image blocks, a similar definition can be made herein: if there exists at least one pixel in both sub-patterns that makes the two adjoin each other within the 4-neighbor domain, these two sub-patterns are neighbors. The basic representation units in SNAM are square, line segment and isolated point. Each isolated point is essentially a square

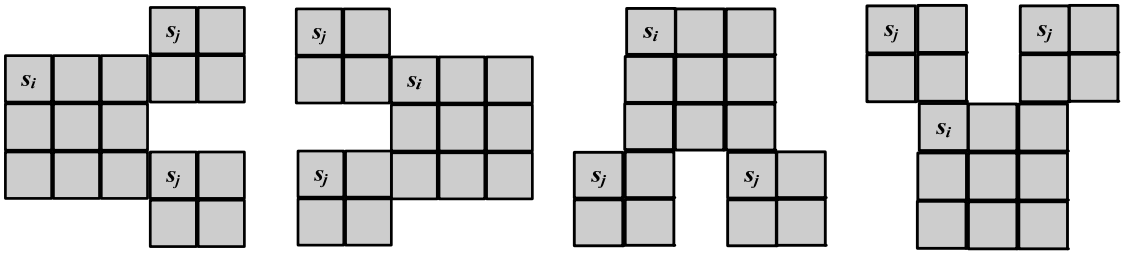
with the side length of 1, thus SNAM actually involves only two types of sub-patterns, namely, square and line segment. The specific definition is as follows:

Definition 1 In the SNAM-based binary image representation, if s_j={x_j,y_j,$Length_j$} neighbors s_i={x_i,y_i,$Length_i$} in the direction D=(EAST,SOUTH,WEST,NORTH):

(1) When s_i is square and s_j is square, their neighbor relationship is shown in Figure 5 and follows the following formula:

$$\begin{cases} x_j = x_i + Length_i \\ y_i - Length_j + 1 \le y_j < y_i + Length_i \end{cases} \quad (1)$$

$$\begin{cases} x_j = x_i - Length_j \\ y_i - Length_j + 1 \le y_j < y_i + Length_i \end{cases} \quad (2)$$



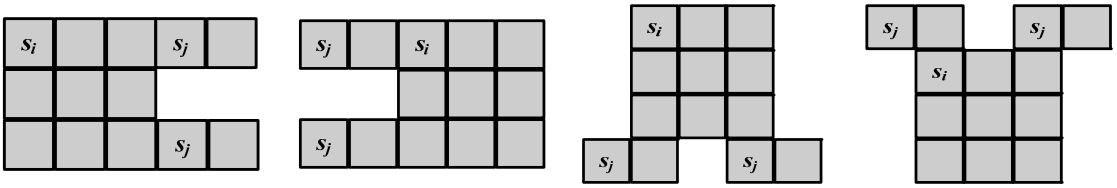
(a) The scope of s_i's east neighbors (b) The scope of s_i's west neighbors (c) The scope of s_i's south neighbors (d) The scope of s_i's north neighbors

Figure 5. The process of anti-packing a binary image with SNAM

$$\begin{cases} x_i - Length_j + 1 \le x_j < x_i + Length_i \\ y_j = y_i + Length_j \end{cases} \quad (3)$$

$$\begin{cases} x_i - Length_j + 1 \le x_j < x_i + Length_i \\ y_j = y_i - Length_j \end{cases} \quad (4)$$

(2) When s_i is square or point and s_j is line segment, their neighbor relationship is shown in Figure 6 and follows the following formula:



(a) The scope of s_i's east neighbors (b) The scope of s_i's west neighbors (c) The scope of s_i's south neighbors (d) The scope of s_i's north neighbors

Figure 6. The neighbor relationship between square s_i and line segment s_j

$$\begin{cases} x_j = x_i + Length_i \\ y_i \le y_j < y_i + Length_i \end{cases} \quad (5)$$

$$\begin{cases} x_j = x_i - Length_j \\ y_i \le y_j < y_i + Length_j \end{cases} \quad (6)$$

$$\begin{cases} x_i - Length_j + 1 \le x_j < x_i + Length_i \\ y_j = y_i + Length_j \end{cases} \quad (7)$$

$$\begin{cases} x_i - Length_j + 1 \le x_j < x_i + Length_i \\ y_j = y_i - 1 \end{cases} \quad (8)$$

(3) When s_i is line segment and s_j is square or point, their neighbor relationship follows the following formula:

$$\begin{cases} x_j = x_i + Length_i \\ y_i - Length_j + 1 \le y_j \le y_i \end{cases} \quad (9)$$

$$\begin{cases} x_j = x_i - Length_j \\ y_i - Length_j + 1 \le y_j \le y_i \end{cases} \quad (10)$$

$$\begin{cases} x_i - Length_j + 1 \le x_j < x_i + Length_i \\ y_j = y_i - Length_j \end{cases} \quad (11)$$

$$\begin{cases} x_i - Length_j + 1 \le x_j < x_i + Length_i \\ y_j = y_i + 1 \end{cases} \quad (12)$$

(4) When s_i is line segment and s_j is line segment, because the minor-diagonal scanning mode only generates horizontal lines which only have south and north neighbors, their neighbor relationship follows the following formula:

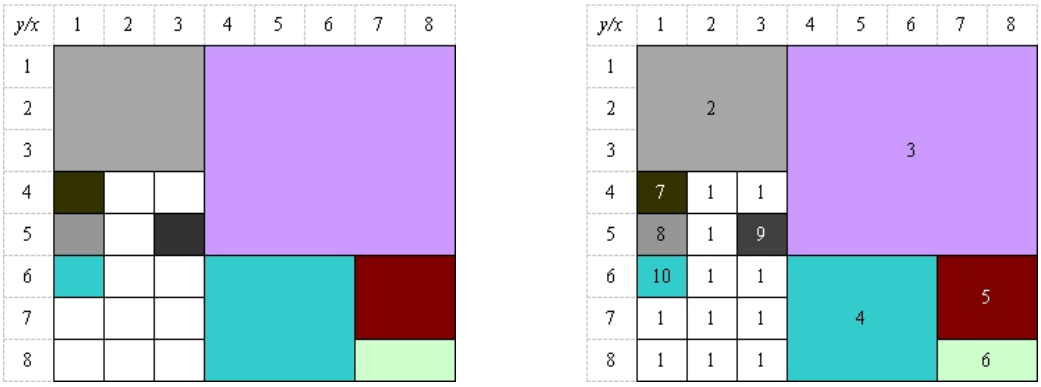
$$\begin{cases} x_i - Length_j + 1 \le x_j < x_i + Length_i \\ y_j = y_i - 1 \end{cases} \quad (13)$$

$$\begin{cases} x_i - Length_j + 1 \le x_j < x_i + Length_i \\ y_j = y_i + 1 \end{cases} \quad (14)$$

3.2. Grid Array

SNAM does not involve the recording of the spatial-position relationship with codes, thus neighbor-finding becomes an inefficient process of repeated definition-based searches

throughout the code array. Therefore, the neighbor-finding process can be simplified by constructing an intermediate structure and restoring the position relationships among sub-patterns. A *n*×*n* matrix, namely grid matrix, can be constructed for a *n*×*n* binary image. Each unit is first assigned with a value of 1; then the SNAM coding result of the image is read, and the position that each sub-pattern corresponds to in the grid matrix, the structure of which is shown in Figure 7, is re-filled with a value, which is equal to "the storage serial number for the corresponding sub-pattern + 2" because the serial number of the SNAM code array begins from 0 and also 1 has been used for pre-filling; and then the SNAM code and the filled value for each sub-pattern make up a grid array, with the filled value acting as the corresponding index. Thus the only step for finding the neighbor of some block is to scan the pixel values surrounding this block - if a different value that is not 1 and detected, the sub-pattern that this index value corresponds to in the grid matrix is the neighbor that is being searching for.



(a) SNAM-based segmentation of an 8×8 image (b) The corresponding grid matrix

Figure 7. The neighbor relationship between line segment s_i and line segment s_j

Without the application of the grid matrix method, neighbor-finding would be a very time-consuming process of traversing the entire SNAM code list to make one-by-one judgment as to whether the currently accessed SNAM sub-pattern meets the condition for neighbor relationship. With grid matrix, the operation scope of neighbor-finding in certain direction can be limited to the pixels on the boundary block of sub-patterns, thus the operation load can be significantly cut.

3.3. Grid Array

After the storage queue *V* of the SNAM algorithm for binary image as well as the V-generated grid matrix *M* and grid array *T* is given, the search for the neighbor *R* of a given sub-pattern *r*= (*x, y, Length*) will become very easy. Take the search for east neighbor for example: first of all, judge the sub-pattern type of *r* according to its *Length*; then determine whether *r* is close to the right boundary of the image according to the abscissa *x+Length* of *r*'s endpoint, and if it is, *r*'s east neighbor *R* is an empty set, otherwise the values for the right boundary of *r* sub-pattern in the grid array are indexes of all its neighbors - then, compare them with the values on the right boundary of the grid array one by one, and add these position-storage indexes to the neighbor set *R* as long as there is no 1-value index. Because each index in the SNAM code list corresponds to one and only sub-pattern, the indexes can lead to the neighboring models. The algorithm principles for neighbor-finding in various directions are basically the same, thus the eastward neighbor-finding is used herein again as an example to explain the specific algorithm execution steps as follows:

Input: grid array *T*, the given coordinates (*x*, *y*) of the starting point of sub-pattern *r*, image resolution n, and code array *V={V_s,V_l,V_p}*, where *V_s, V_l* and *V_p* respectively represent the code array set for square, line segment and isolated point.

Output: the set *E_R* of sub-patterns of *r*'s east neighbors

Step 1 Customize a set *E_R* and initialize it to be empty.

Step 2 Input the given coordinates (*x*, *y*) of the starting point of sub-pattern *r* that is used for neighbor-finding, and match the starting point's coordinates with the code array to produce the basic information and serial number (index) of sub-pattern *r*.

Step 3 Use the basic information of the sub-pattern to judge the basic type of it: if it is square or isolated-point, jump to Step 4; if it is line segment, which means it has no east or west neighbors, jump to Step 5.

Step 4 Via the coordinates (*x*, *y*) of *r*'s starting point, find *r*'s right-most endpoint *(x+length,y)* in the grid matrix *M*. If *r* is close to the right boundary of the image, that is, *x+length* is equal to n, then it does not have east or west neighbors, thus go to Step 5; otherwise, thoroughly scan M for pixel units located within the area (*x+length*+1, y+*length*-1) on the right side of the image' right boundary, and judge whether its numerical value (index) m is 1 (representing white point): if it is not 1, use m to find the corresponding sub-pattern in the grid array *T*, and record it in *E_R*.

Step 5 Output the *E_R*.

4. Experiments and Analysis

The experimental environment is made up by Intel Core i3-3110M 2.4GHz processor, 2G memory, Microsoft Windows 7 Professional + SP3 operating system and MATLAB7.0-based IDE. Figure 8 shows eight 256×256 binary images with different complexities to be tested in the experiment. The definition of image complexity is as follows:

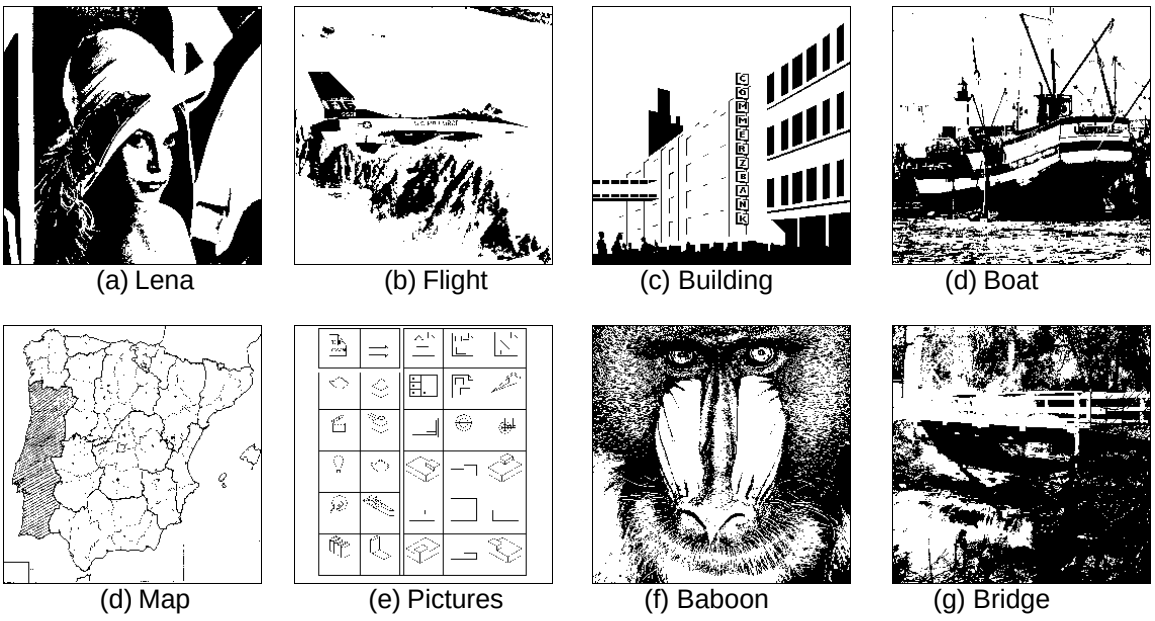
Definition 2 Complexity *C* of any binary image can be defined in the following way: calculate, based on the given image representation method, the total number of its nodes N_Q as well as the total pixel number N_f, and put them in the formula of:

$$C = N_Q / N_f \quad (15)$$

In this paper, the LQT representation method is regarded as a benchmark for comparison, and image complexity is measured by the number of LQT segmentation blocks. Table 1 is the comparison of the coding performances of LQT, SNAM and MD-SNAM. Therein, N represents the number of sub-patterns or nodes, SNAM is the original square NAM-based representation method, MD-SNAM is SNAM based on the minor-diagonal scan, η_{LQT_T} is the ratio of the total data amount of LQT to that of TNAM, η_{LQT_R} is the ratio of the total data amount of LQT to that of RNAM, η_{LQT_S} is the ratio of the total data amount of LQT to that of SNAM, and $\eta_{LQT\_MD-SNAM}$ is the ratio of the total data amount of LQT to that of MD-SNAM. The specific data in the experimental result is shown in Figure 8. Table 1 shows that, for the above eight images, SNAM and MD-SNAM have outperformed LQT, though these two also have significant differences in performance.

The original SNAM needs three fields - coordinates of the starting point and side length - for recording square, four fields - coordinates of the starting point and of the endpoint - for recording line segment, and only 2 fields, namely the point coordinates, for recording isolated point. By comparison, MD-SNAM provides a uniform storage structure for all these types: for recording square, it is the same as SNAM; because it only generates horizontal lines, it only needs three fields - coordinates of the starting point and the length - for recording line segment; and it also needs three fields for recording isolated point, which is regarded as square with a side length of 1. Obviously, MD-SNAM involves one less field than the original SNAM for line segment, but one more field for isolated point. After coding, the numbers of line segments and isolated points vary form one image to another, leading to the difference in the data amount in the coding result. Therefore, the difference in storage structure actually does not have a decisive impact on the coding performances of MD-SNAM and SNAM. Furthermore, compared with the original SNAM, MD-SNAM not only has an optimized storage structure, and also adopts an improved scanning mode - namely, the minor-diagonal scanning mode, providing it with a stronger adaptation to images' local blocky textures, so as to increase squares and decrease nodes. According to the experimental result, regardless of image complexity, MD-SNAM has

fewer nodes, higher compression ratio and better spatial efficiency than the original SNAM, thus it can further reduce the time complexity of the subsequent image processing.



(a) Lena (b) Flight (c) Building (d) Boat

(d) Map (e) Pictures (f) Baboon (g) Bridge

Figure 8. Eight 256 x 256 binary images

Table 1. Comparison of the performances of LQT, SNAM, MD-SNAM algorithms

Image	C	N			η_{LQT_T}	η_{LQT_S}	$\eta_{LQT\_MD-SNAM}$
		LQT	SNAM	MD-SNAM			
Map	0.0882	5780	2708	1090	3.9609	4.2647	10.1613
Pictures	0.0835	5470	742	459	12.9781	16.0064	22.8413
Baboon	0.2601	17047	8668	4204	3.1227	3.8013	7.7719
Bridge	0.1813	11881	6057	3411	2.8796	3.7987	6.6760
Lena	0.0981	6430	3077	2466	2.9902	4.0376	4.9976
Flight	0.0850	5568	2253	1946	3.5287	4.7537	5.4840
Building	0.0702	4602	1425	1728	4.0114	6.0236	5.1044
Boat	0.1313	8604	3515	2731	3.3743	4.6094	6.0384

For square and line segment, MD-SNAM gives different definitions of neighbor, but the principles of searching for their neighbors in the grid matrix are identical. In fact, the operation amount of finding the neighbor of a line segment, which is 1 pixel in width, is less than that of finding a square's neighbor. Since LQT's basic representation unit is square image block, a square sub-pattern is chosen as a benchmark for MD-SNAM in the experiment. The experiment is aimed at comparing the performances in an eastward neighbor-find process, and the result is shown in Table 2, where, T_{LQT} and $T_{MD\text{-}SNAM}$ represent the execution time of neighbor-finding based respectively on LQT and on MD-SNAM, and $t_{L\text{-}MS}$ represents the speed-up ratio of MD-SNAM-based neighbor-finding to LQT-based neighbor-finding.

It can be seen from the Table 2 that higher image complexity can result in longer execution of the eastward neighbor-finding with the two algorithms, indicating that image complexity is proportional to the execution time of the corresponding algorithm. The average execution time of LQT-based neighbor-finding is 4.275~13.716 times that of neighbor-finding with SNAM based on grid array, which fully demonstrates that the neighbor-finding algorithm based on MD-SNAM is simpler, faster, and more supportive for other subsequent image processing.

Table 2. Comparison of the execution time of an eastward neighbor-finding process respectively with the LQT-based algorithm and with MD-SNAM-based algorithm

Image	C	Execution time (ms)		Speedup ratio
		T_{LQT}	$T_{MD\text{-}SNAM}$	$t_{L\text{-}MS}$
Map	0.0882	18.764	1.368	13.716
Pictures	0.0835	17.976	1.872	9.602
Baboon	0.2601	24.653	3.896	13.190
Bridge	0.1813	22.889	2.769	9.930
Lena	0.0981	19.774	1.608	5.611
Flight	0.0850	18.991	1.347	4.275
Building	0.0702	17.437	1.246	4.552
Boat	0.1313	20.988	2.157	6.581

5. Conclusion

This paper, firstly, improves the scanning mode of the original SNAM so that it can handle images with various texture characteristics; secondly, optimizes the storage structure for sub-patterns so as to eliminate the storage difference in various sub-patterns; thirdly, realizes the description of the position relationships among sub-patterns with the grid array; and finally, uses the grid array to limit the range of one-directional neighbor-finding to the boundary of the grid array that the benchmark blocks correspond to, thus creating a new neighbor-finding algorithm, which has higher execution efficiency than the classic LQT-based method and can be applied in image operations such as calculation of geometric properties and moment generation.

Acknowledgements

This work was supported by Guangxi Natural Science Foundation Program (Grant No. 2013GXNSFBA019275, Grant No. 2013GXNSFBA019276), Guangxi University of Science and Technology Research Program (Grant No. 2013YB227, Grant No. 2013YB228) and National Natural Science Foundation of China (Grant No: 61202464).

References

[1] Xing Z, Shui L. Image Edge Feature Extraction and Refining Based on Genetic-Ant Colony Algorithm. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*. 2015; 13(1): 118-127.
[2] JJ Laguardia, E Cueto, M Doblare. A Natural Neighbor Galerkin Method with Quadtree Structure. *International Journal for Numerical Methods in Engineering*. 2005; 63(6): 789-812.
[3] I Gargantini. An Effective Way to Represent Quadtrees. *Communications of the ACM*. 1982; 25(12): 905-910.
[4] V Khanna, P Gupta, JC Hwang. Maintaining Connected Components in Quadtree-Based Representation of Images. *Iranian Journal of Electrical and Computer Engineering*. 2003; 2(1): 53-60.
[5] BP Kumar, P Gupta, CJ Hwang. An Efficient Quadtree Data Structure for Neighbor Finding Algorithm. *International Journal of Image and Graphics*. 2001; 1(4): 619-633.
[6] P Bhattacharya. Efficient Neighbor-Finding Algorithm in Quadtree and Octree. Kanpur: Indian Institute of Technology, 2002.
[7] S De Bruin, W De Wit, J Van Oort, et.al. Using Quadtree Segmentation to Support Error Modeling in Categorical Raster Data. *International Journal of Geographical Information Science*. 2004; 18(2): 151-168.
[8] CL Wang, SC Wu, YK Chang. Quadtree and Statistical Model-Based Lossless Binary Image Compression Method. *Imaging Science Journal*. 2005; 53(2): 95-103.
[9] G Minglun, Y Yee-Hong. Quadtree-Based Genetic Algorithm and Its Applications to Computer Vision. *Pattern Recognition*. 2004; 37(8): 1723-1733.
[10] Chuanbo C. Study on A Non-Symmetry and Anti-Packing Pattern Representation Method. Wuhan: Huazhong University of Science and Technology, 2005.
[11] Peng W, Chuanbo C, Yunping Z. Image Set-Operation Algorithm based on NAM and Its Experimental Research. *Journal of Yangtze University (Natural Science Edition)*. 2009; 6(2): 76-79.
[12] Weiju H. Study on Multiple Sub-Pattern Image Representation and Retrieve Methods based on NAM. Wuhan: Huazhong University of Science and Technology, 2009.

[13] Dehong Q, Xinxin P, Chuanbo C, Shaohong F. Sequence Classification based on Feature Subsequence Optimized through Semidefinite Program. *Journal of Chinese Computer Systems*. 2008; 29(11): 2083-2086.
[14] Junjie P. Graph-based NAM Representation and Salient Region Detection. Wuhan: Huazhong University of Science and Technology, 2011.
[15] Jie H, Hui G. A Square NAM Representation Method for Binary Images. *Applied Mechanics and Materials*. 2012; 133-134: 755-759.
[16] Bo W, Yubin G. An Image Compression Scheme Based on Fuzzy Neural Network. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*. 2015; 13(1): 137-145.
[17] Yunping Z, Chuanbo C. An Optimization Strategy of NAM using Raster Scanning. *Journal of Huazhong University of Science and Technology (Natural Science Edition)*. 2008; 36(8): 1-4.
[18] BP Kumar, P Gupta, CJ Hwang. An Efficient Quadtree Data structure for Neighbor Finding Algorithm. *International Journal of Image and Graphics*. 2001; 1(4): 619-633.