A Neighbor Searching Algorithm for Gray Images Based on SNAMG Representation

Jie He, Hui Guo

Guangxi Colleges and Universities Key Laboratory of Professional Software Technology, Wuzhou University, Wuzhou 543002, Guangxi, China

Abstract

Aiming at the problems that the neighbor searching method based on pixels can only be executed to binary images with low efficiency, the paper strengthened texture adaptability through an optimized anti-diagonal scanning manner based on the SNAMG image representation method; then brought in matching thresholds to reduce total quantity of square-represented units through application of human visual system characteristics; and finally put forward a simplified neighbor searching algorithm for gray images and derivative algorithms for perimeter and area calculation. It is shown in experimental results that these algorithms can be adapted to gray images with different complexities and obtain high execution efficiency.

Key words: SNAMG IMAGE REPRESENTATION METHOD, ANTI-DIAGONAL SCANNING MANNER, HUMAN VISUAL SYSTEM CHARACTERISTICS, NEIGHBOR SEARCHING

1. Introduction

Neighbor searching algorithm is the foundation of geometrical calculation of images, image measurement, labeling of connected domains, edge detection and many other image processing courses [1]. The representation method adopted in digital images and its neighbor definition decides performances of the neighbor searching algorithm and space-time complexity. The original neighbor searching algorithm searches bitmap images through pixel-pixel traversal searching manner with very low efficiency. The widely applied quadtree method [2] and linear quadtree [3] method reduce the quantity of basic representation units of images, save storage space and realize quick execution of image calculation in the "block-block" manner. Literatures [4-6] respectively put forward neighbor searching algorithm s based on quadtree or linear quadtree representation, and played a very important role in increasing efficiency of some image processing calculations [7-9]. However, a neighbor relationship is not a one-to-one relation, so that methods of this type are complex in actual image operations. Hence, with continuous appearance of new and simpler image representation methods, people are still looking for efficient neighbor searching methods.

In comparison with the quadtree method, the non-symmetry and anti-packing image representation method [10] has higher representation efficiency and derives a series of

efficient image processing algorithms [11-14]. Literature [15] put forward a gray image representation method by taking squares, isolated points, line segments and the like as basic representation units of NAM method (Square non-symmetry and anti-packing model for gray images, SNAMG). These basic representation units have normalized shapes and simple structures, so that it is feasible to take them as the foundation for further development of other image operation algorithms. However, the raster scanning manner in this method can easily generate a lot of sub-patterns during image coding with abundant massive textures and complex gray scale distribution, which restrain the supporting of SNAMG method to other image processing courses.

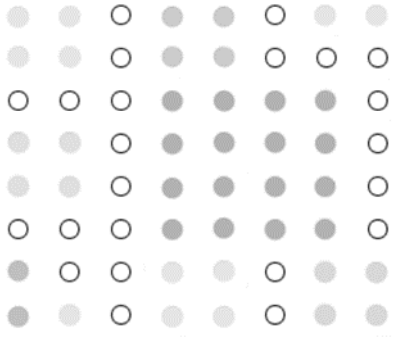
Efficiency of the neighbor searching algorithm depends on total quantity of basic representation units. Hence, reduction of quantity of sub-patterns is the key factor for efficient neighbor searching in SNAMG. At first, the author firstly made use of structural characteristics of squares, added sub-scanning in the anti-diagonal direction to the raster scanning, and strengthened the adaptability to massive textures during anti-packing; afterwards, the author introduced threshold segmentation through application of human visual characteristics, expanded square coverage, reduced quantity of line segments and points, and further simplified data size. By above measures, the author realized efficient neighbor searching and calculation of perimeter and area in SNAMG coding results.

2. SNAMG gray image representation method and its optimization

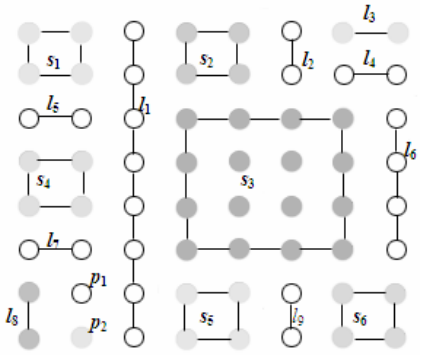
2.1. Square non-symmetry and anti-packing representation method of gray images

SNAMG gray image representation method is a lossless representation method established in spatial domain. Three sub-patterns including squares, line segments and isolated points are taken as basic image representation units. According to the raster scanning manner, squares, line segments and isolated points with different scales are extracted from a packed gray image by the anti-packing algorithm. After that, corresponding parameters of these sub-patterns with different scales are recorded and stored to represent the given gray image.

Figure 1 shows results about anti-packing coding and storage of a gray image with application of the SNAMG method. Figure 1(a) gives an original gray image T with size of 8×8 and gray scale of 8. Figure 1(b) shows results of SNAM anti-packing with application of the raster scanning method, wherein 6 squares s1—s6, 9 line segments l1—l2 and 2 isolated points p1, p2 were extracted from the original diagram.



(a) Gray image G



(b) SNAMG anti-packing results of G

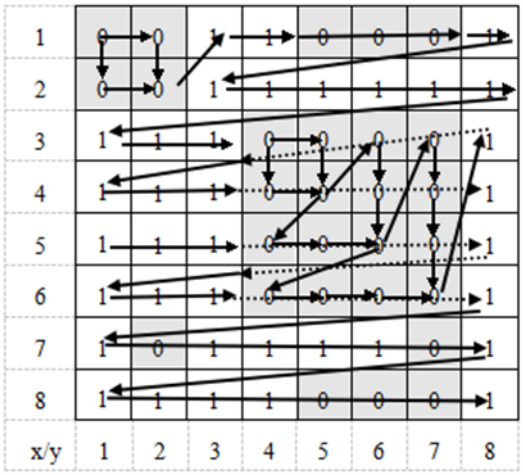
Figure 1. Principles of SNAMG coding

Some problems going against image operations still exist in the SNAMG representation method. Literature [16] pointed out that the raster scanning method could show good performances when massive characteristics of image textures were not obvious; and showed low efficiency in representation of images with obvious massive characteristics. In addition, 3 sub-patterns of SNAM had different storage structures; its original anti-packing algorithm led to line segments in horizontal and vertical directions among the coding results. All these consequences will make subsequent image operations more complex.

2.2. Optimization of scanning manner

Raster scanning is characterized in that scanning is carried out row by row in the horizontal direction. If the massive scanning manner can be combined based on it, the applicability to massive texture images can be strengthened. Hence, the principal scanning direction is still the horizontal direction. However, when unlabeled black points are scanned, the anti-diagonal scanning manner can be developed by applying structural characteristics of square and taking the judgment of pixel values of anti-diagonal pixel points as the algorithm core. With a

binary image as the example, its principles are shown in Figure 2. Figure2(a) shows the direction of anti-diagonal scanning, wherein the imaginary line indicates that these points need not to be scanned in the principal horizontal direction after they form a square. Figure 2(b) shows serial numbers of scanned sequence of each pixel point during anti-diagonal scanning. Besides stronger adaptability, such manner will generate line segments only in the horizontal direction, reducing complexity of the SNAM method.



(a) Direction of anti-diagonal scanning

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) Sequence of pixel points in anti-diagonal scanning

Figure 2. Principles of anti-diagonal scanning

2.3. Optimization of segmentation manner

Each pixel point of a gray image has a gray scale value, so that its neighbor searching and calculation of geometrical properties become complex. Human visual system can tolerate big quantization errors in the image edge areas and cannot completely distinguish all textures and gray scale changes in the image. Hence, without influencing visual impressions, the image representation method can be simplified by setting segmentation thresholds. As for the SNAMG method, gray scale value g of the square initial point during matching search is taken as the standard reference value. The threshold is set to be x, so that the pixel points with gray scale values in the interval of g+/-(x/2) during square matching can be included into the same square. In this way, line segments and isolated points can be greatly reduced.

50	50	25	28	28
51	47	25	28	28
15	14	15	53	52
15	15	14	53	51
16	16	15	100	101

Figure 3. No Threshold

50	50	25	28	28
51	47	25	28	28
15	14	15	53	52
15	15	14	53	51
16	16	15	100	101

Figure 4. Threshold = 5

As shown in Figure 3, if the threshold is not set during segmentation, a lot of line segments and isolated points will be formed, making further storage and calculation of image more complex. As shown in Figure 4, by setting the threshold, a lot of line segments and points can be included into the square for representation as much as possible, so that only square-related properties need to be considered during neighbor searching of the gray image.

2.4. Algorithm description

Figure 5 shows the square storage structure obtained by the above improved SNAMG method. Therein, *sp* indicates vertex coordinates of top left corner of the square, length indicates side length of the square, and z indicates gray scale value of the square.

sp	*Length*	*z*

Figure 5. Storage Structure of Square Sub-pattern

Specific coding steps are as follows:
Input: gray image G with single image size of 2n×2n and bit depth of m.

Output: square sub-pattern set V_s {s_num}.

Step 1: A proper threshold g is set for the gray image. The gray image is scanned by the anti-diagonal scanning manner. At first, the pixel point of the first unlabeled random gray scale value is searched and the gray scale value is taken as the standard gray scale value. After that, neighbor points of the pixel point are searched by the SNAM anti-packing algorithm. If the neighbor gray scale values are within the threshold scope, they will then be matched until a square with the maximum side length is formed. All the pixel points forming the square are labeled and these labeled pixel points are converted into the standard gray scale value.

Step 2: Vertex coordinates of top left corner, side length and gray scale value of the square sub-pattern are recorded. Its two-dimensional coordinates (x, y) are converted into one-dimensional coordinates sp by dimensionality reduction.

Step 3: sp, length and the gray scale value z are stored into the set V of square sub-patterns, namely V= {s_num} ← {sp, length, z}. Therein, sp indicates coordinates of top left corner of the square, length indicates side length of the square, and z indicates the gray scale value corresponding to the square.

Step 4: Step 1 and Step 2 are circulated to continue searching the rest square sub-patterns until no square sub-pattern can be found.

Step 5: the square sub-pattern set V {s_num} of the anti-packing coding queue of G is output.

3. Neighbor searching based on SNAMG

3.1. Definition of neighboring

Neighboring can be used to describe the neighboring relations among image areas corresponding to image representation units. Its definition relies on a specific image representation method. Literature [18] proposed the definition of neighboring based on the quadtree method. Both the SNAMG representation and the quadtree representation describe a source Figureure into a recoding set of partial image blocks. Hence, the two kinds of representation can adopt a similar definition method: each of the two sub-patterns at least has one pixel which can make the two sub-patterns mutually neighbor in 4 neighborhoods, so that they have a neighboring relationship. Basic representation units of SNAMG contain 3 sub-patterns: squares, line segments and isolated points. In essence, an isolated point is a square with side length of 1, so that SNAMG indeed can be deemed as two sub-patterns – squares and line segments. After introducing anti-diagonal scanning and matching threshold, neighbor searching based on SNAMG only needs to judge neighboring among relevant square sub-patterns.

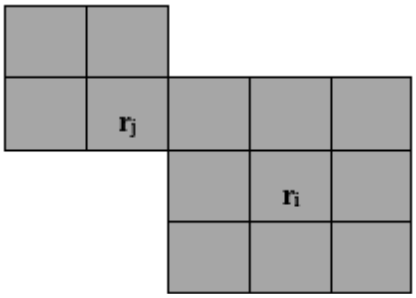
Definition 1: Neighboring Relations of SNAMG Using a Threshold: if a square $r_j = \{x_j, y_j, Length_j\}$ is neighbor to the square $r_i = \{x_i, y_i, Length_i\}$ in a direction D = (EAST, SOUTH, WEST, NORTH), the following formula shall be satisfied, as shown in Figure 6.

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

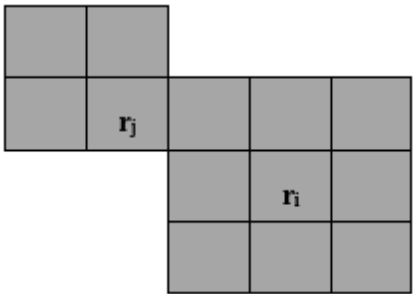
$$\begin{cases} x_j = x_i - Length_j \\ y_i - Length_j + 1 \le y_j < y_i + Length_i \end{cases} \quad (2)$$

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

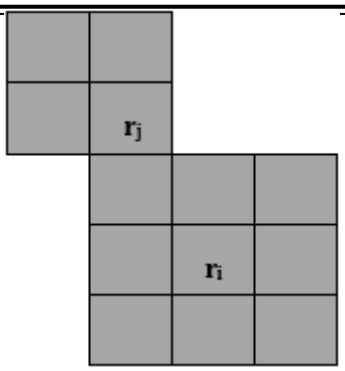
$$\begin{cases} x_i - Length_j + 1 \le x_j < x_i + Length_i \\ y_j = y_i - Length_j \end{cases} \quad (4)$$



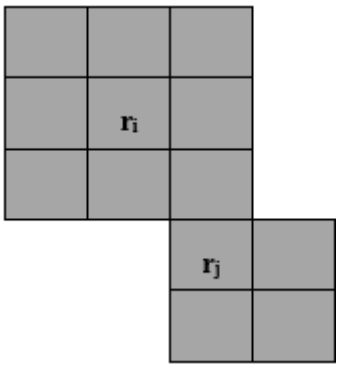
(a) r_i has a top right neighbor r_j



(b) r_i has a left neighbor r_j



(c)r_i has a top neighbor r_j



(d)r_i has a below neighbor r_j

Figure 6. Several Kinds of Neighboring among Square Sub-patterns

It is shown in above diagram that if two square sub-patterns can be connected by 4 neighborhoods of a pixel of each sub-pattern, the two squares are deemed to be neighbor.

3.2 Principles of algorithm

During storage through application of anti-diagonal scanning, a matching threshold is set for the gray image, pixel points within the threshold scope are included into the same square, and the rest line segments and points are scanned and stored at the same time; after that, total quantity s_num of square sub-patterns, total quantity l_num of line segments and total quantity p_num of points are calculated, vertex coordinates, gray scale values and side lengths of squares, line segments and isolated points are stored into corresponding coding queues.

The decoding filling algorithm is the key for realization of neighbor searching. A universal matrix with size of n×n is established, vertex coordinates, lengths and gray scale values are then extracted from the queue storing squares, and corresponding positions in the matrix are filled by the gray scale values. After filling of all the squares, the left positions in the matrix are all filled by -1. Whether 4 neighborhoods of each pixel point on square edges are -1 is judged. If not, the square is deemed to have a neighbor square, and then the gray scale value of the neighbor square is recorded. At last, position of the neighbor square existing in the 4 neighborhoods of the square as well as the corresponding gray scale value are output.

3.3 Description of algorithm

Specific steps of neighbor searching algorithm based on SNAMG are as follows:

Input: a universal matrix of n×n, the starting point coordinates (x, y) of a given sub-pattern r for neighbor searching, image resolution ratio n, and coding queue square= {s_num}, wherein s_num indicates the set of coding queues of squares.

Output: set R of square neighbors of r in different directions.

Step 1: a set R is customized, and it is initialized into empty.

Step 2: starting point coordinates (x, y) of the given square sub-pattern r for neighbor searching are input, and position, side length, gray scale value and other basic information of r are obtained by matching the initial coordinates with the coding queue.

Step 3: whether a neighbor square exists in top, bottom, left and right neighborhoods of the square sub-pattern is judged by basic information of the square r.

Step 3.1: whether a neighbor exists on the right side is judged. By initial coordinates (x, y) of r, coordinates (x+length-1, y) of the most right end are calculated. If the coordinates are equal to those of the image right edge, namely x+length-1 = n, the operation will be directly skipped to Step 3.2. Otherwise, all the pixel units from the right side (x+length, y) of the r right edge to (x+length, y+length-1) will be scanned. Whether their corresponding values are -1 is judged. If not, a right neighbor square is deemed to exit. Its gray scale value, coordinates and side length are obtained and recoded in R.

Step 3.2: whether a neighbor exists on the bottom side is judged. By initial coordinates (x, y) of r, coordinates (x, y+length-1) of the most bottom end of *r* are obtained. If they are equal to those of bottom edge of the image, namely y+length-1 = n, the operation will be directly skipped to Step 3.3. Otherwise, all the pixel units from the bottom side (x, y+length) of the r edge to (x+length-1, y+length) will be scanned. Whether their corresponding values are -1 is judged. If not, a bottom neighbor square is deemed to exit. Its gray scale value, coordinates and side length are obtained and recoded in R.

Step 3.3: whether a neighbor exists on the left side is judged: by initial coordinates (x, y) of

r, coordinates (x,y) of the most left end of *r* are obtained. If they are equal to those of left edge of the image, namely x<1, the operation will be directly skipped to Step 3.4. Otherwise, all the pixel units from the left side (x-1, y) of the r left edge to (x-1, y+length-1) will be scanned. Whether their corresponding values are -1 is judged. If not, a left neighbor square is deemed to exit. Its gray scale value, coordinates and side length are obtained and recoded in R.

Step 3.4: whether a square sub-pattern exists on the top side is judged: by initial coordinates (x, y) of r, coordinates (x, y) of the most top end of r in the matrix are obtained. If they are equal to those of top edge of the image, namely y<1, the operation will be directly skipped to Step 4. Otherwise, all the pixel units from the top side (x, y-1) of the r top edge to (x+length-1, y-1) will be scanned. Whether their corresponding values are -1 is judged. If not, a top neighbor square is deemed to exit. Its gray scale value, coordinates and side length are obtained and recoded in R.

Step 4: R is output

4. Perimeter calculation based on SNAMG

4.1. Definition of perimeter and algorithm principles

It is assumed that squares with quantity of s_num are stored in the coding queue. Hence, perimeter calculation can also be realized by a decoding filling matrix. A universal matrix with size of n×n is set up. Vertex coordinates and length are extracted from the square storage queue. According to vertex coordinates and side length of the square, the covered matrix scope is determined. In order to realize more convenient calculation, scopes covered by each square are in sequence assigned as integers from 2 to s_num. In this way, 4 neighborhoods and 8 neighborhoods of edge pixels of each square can be judged in order to calculate perimeter of a gray image, as shown in Figure 7.

Edges of some squares have coincided with an edge of the original image, so that there may be no pixel beside them and it is hardly to calculate length of the edge by above methods. In order to solve this problem, a row or a line of pixel units with value of 1 can be added respectively to 4 edges of the original filling matrix, so that the matrix can be expanded to a (n+2)×(n+2) matrix, as shown in Figure 8. If quantity of edge pixel points of 8 neighborhoods which pass edge pixels of the judgment area is equal to perimeter of the area, the perimeter C of a square sub-pattern can be defined as follows:

$$C = \sum_{i=1}^{n} \sigma_8 \qquad (5)$$

y\x	1	2	3	4	5	6	7	8
1	2	2	2	1	3	3	3	3
2	2	2	2	1	3	3	3	3
3	2	2	2	1	3	3	3	3
4	1	4	4	4	3	3	3	3
5	1	4	4	4	1	1	6	6
6	1	4	4	4	5	5	6	6
7	1	1	1	1	5	5	1	1
8	1	1	1	1	1	1	1	1

Figure 7. Assignment Filling Matrix

y\x	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1	1
2	1	2	2	2	1	3	3	3	3	1
3	1	2	2	2	1	3	3	3	3	1
4	1	2	2	2	1	3	3	3	3	1
5	1	1	4	4	4	3	3	3	3	1
6	1	1	4	4	4	1	1	6	6	1
7	1	1	4	4	4	5	5	6	6	1
8	1	1	1	1	1	5	5	1	1	1
9	1	1	1	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1	1	1	1

Figure 8. Expanded Assignment Filling Matrix

4.2 Description of algorithm

The algorithm steps for gray image perimeter (8 neighborhoods) calculation based on SNAMG are as follows:

Input: a universal matrix T of n×n, a square coding queue Square = {s_num} of a gray image G.

Output: perimeter C of G.

Step 1: C is initialized into 0, T is expanded to a matrix T' of (n+2) × (n+2).

Step 2: if Square is empty, the operation is skipped to Step 5.

Step 3: square sub-patterns in Square are extracted in sequence and filled in T'. The coverage area of each square is assigned as integers within (2, s_num) according to the extraction sequence.

Step 4: according to basic information of square sub-pattern, whether points with pixel of 1 exist in the 8 neighborhoods of the square sub-pattern is judged.

Step 4.1: top edge of the square sub-pattern is judged. By initial coordinates (x, y) of r, all elements from the square top edge (x, y) to (x+length-1, y) are traversed. If points with pixel of 1 exist in the 8 neighborhoods of each pixel corresponding to the top edge, C=C+1 and the operation will be skipped to Step 5.

Step 4.2: bottom edge of the square sub-pattern is judged. By initial coordinates (x, y) of r, all elements from the square bottom edge (x, y+length-1) to (x+length-1, y+length-1) are traversed. If points with pixel of 1 exist in the 8 neighborhoods of each pixel corresponding to the bottom edge, C=C+1 and the operation will be skipped to Step 5.

Step 4.3: left edge of the square sub-pattern is judged. By initial coordinates (x, y) of r, all elements from the square left edge (x, y) to (x, y+length-1) are traversed. If points with pixel of 1 exist in the 8 neighborhoods of each pixel corresponding to the left edge, C=C+1 and the operation will be skipped to Step 5.

Step 4.4: right edge of the square sub-pattern is judged. By initial coordinates (x, y) of r, all elements from the square right edge (x+length-1, y) to (x+length-1, y+length-1) are traversed. If points with pixel of 1 exist in the 8 neighborhoods of each pixel corresponding to the right edge, C=C+1 and the operation will be skipped to Step 5.

Step 5: C is output.

5. Area calculation based on SNAMG

5.1. Definition of area and algorithm principles

Definition 2: it is assumed that G is a gray image and area of a domain block determined by a single square sub-pattern coded through SNAM is S, so that the area of the domain block determined by the single square sub-pattern can be determined by the following formula.

$$Area = length * length \quad (6)$$

Definition 3: it is assumed that after SNAM coding, the gray image M has square sub-patterns with total quantity of t, so that the area S of the gray image can be determined by the following formula.

$$S = \sum_{i=1}^{t} Area_i \quad (7)$$

5.2. Description of algorithm

Input: square coding queue Square = {s_num} of gray image G

Output: Area of G.

Step 1: Area is initialized to be 0.

Step 2: if Square is empty, the operation is skipped to Step 5.

Step 3: square sub-patterns in V are extracted in sequence to obtain the side length length of square sub-pattern.

Step 4: Area=Area + $(length)^2$, wherein $(length)^2$ indicates the area of the region determined by the square sub-pattern.

Step 5: Area is output.

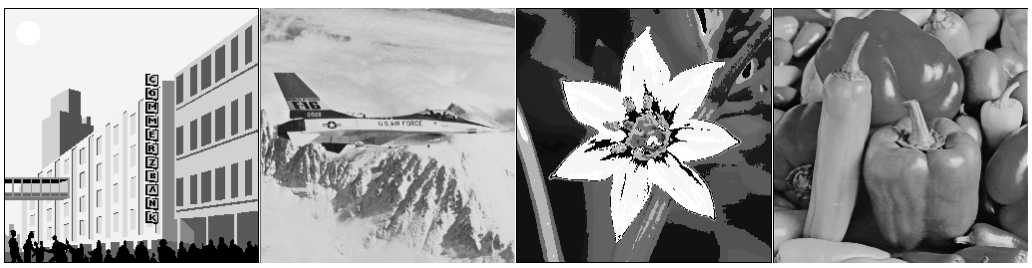
6. Experiment and analysis

The experiment was carried out by Windows 7.0, Core i3 2.3G processor, 2G internal storage, and Matlab7.0 integrated development tool. Different experimental images were selected with image complexity as the measurement index. Image complexity reflects the complexity degree of image textures and can indirectly reflect adaptability of an algorithm. Hence, quantity of coded squares in the widely applied classic LQT algorithm is taken as the standard to define the image complexity.

Definition 4: Complexity of Gray Image: complexity C of any gray image can be defined by the following formula.

$$C=N_{LQT}/N_f \quad (8)$$

Where: N_{LQT} indicates quantity of squares obtained when the gray image is represented by LQT, N_f indicates sum of all the pixels of the image. Obviously, $0<C\leq 1$.



(a)Building (b) Flight (c) Flower (d) Peppers

Figure 8. 4 Gray Images

Experimental images are shown in Figure 8. They are all gray images of 256×256. Table 1 shows relevant data about complexity corresponding to each image, wherein *N* is sum of all the pixels, N_{LQT} indicates quantity of squares

generated from LQT coding, and *Cg* is the image complexity.

Table 1. Complexity of each experimental image

Image	*N*	N_{LQT}	*Cg*
Building	65536	16351	0.2495
Flight	65536	65071	0.9929
Flower	65536	30073	0.4589
Peppers	65536	65452	0.9987

It is shown in Table 1 that 4 images had different complexities, wherein quantities of LQT coded squares of Flight and Peppers approached the sum of all the pixels, and their complexities approached 1, indicating that gray scale distribution of their pixels were extremely disperse and complex. Complexity of Building was only 0.2495, indicating that its gray scale distribution was relatively concentrative and simple. Table 2-Table 7 respectively show performances of algorithms such as SNAMG and its neighbor searching and perimeter calculation in different images under different thresholds. Therein, *Threshold* indicates a matching threshold used during coding segmentation, N_S indicates quantity of squares generated from coding, *T* indicates coding duration, T_{NB} indicates the average neighbor searching duration obtained by testing over 100 squares in the SNAMG coding results, *PSNR* indicates the peak signal to noise ratio of a decoded image, C indicates the sum of perimeters of 8 neighborhoods of all the squares in the image, T_C indicates average duration for execution of perimeter algorithm, S indicates the sum of areas of all the squares in the image, T_S indicates the execution duration of area algorithm, the unit of *T*, T_{NB}, T_C and T_S is second, and unit of *PSNR* is dB.

Table 2. Performances of SNAMG coding, neighbor searching, perimeter and area calculation of 'Building' under different thresholds

Threshold	N_S	*T*	T_{NB}	*PSNR*	C	T_C	*S*	T_S
0	3524	0.827	0.016	Inf	9164	0.061	42228	0.015
10	3429	0.874	0.016	34.5184	9190	0.050	47066	0.015
20	3429	1.216	0.047	34.5184	9190	0.015	47066	0.015
30	3276	1.264	0.046	34.3579	9004	0.004	54756	0.014

Table 3. Performances of SNAMG coding, neighbor searching, perimeter and area calculation of 'Flight' under different thresholds

Threshold	N_S	*T*	T_{NB}	*PSNR*	C	T_C	*S*	T_S
0	383	1.373	0.031	Inf	1568	0.006	415	0.001
10	4919	1.139	0.031	43.1182	17956	0.075	19071	0.016
20	4194	0.967	0.047	37.3891	14572	0.062	28911	0.016
30	3935	0.951	0.047	34.3813	13622	0.065	33517	0.015

Table 4. Performances of SNAMG coding, neighbor searching, perimeter and area calculation of 'Flower' under different thresholds

Threshold	N_S	*T*	T_{NB}	*PSNR*	C	T_C	*S*	T_S
0	4889	1.077	0.063	Inf	20598	0.079	25533	0.016
10	4627	1.139	0.031	43.1182	19591	0.067	30693	0.016
20	3371	0.967	0.047	37.3891	12732	0.054	40146	0.014
30	3295	0.951	0.047	34.3813	12797	0.056	50304	0.014

Table 5. Performances of SNAMG coding, neighbor searching, perimeter and area calculation of 'Peppers' under different thresholds

Threshold	N_S	*T*	T_{NB}	*PSNR*	C	T_C	*S*	T_S
0	77	1.170	0.015	Inf	308	0.007	77	0.001
10	6093	1.451	0.031	42.7286	24158	0.095	14277	0.031
20	5877	1.107	0.046	36.4947	19022	0.089	25419	0.031
30	5159	1.138	0.046	33.2577	15679	0.086	31872	0.016

It is thus clear by combining Table 1 and Table 2-Table 5 that 4 images had different complexities, wherein quantities of LQT coded squares of Flight and Peppers approached sum of all the pixels, and their complexities approached 1, indicating that their pixel gray scale distribution was extremely disperse and complex. Hence, when SNAMG coding was used under the threshold of 0, there were very few squares in the coding results, so that sum of perimeters, sum of areas, and execution duration of each algorithm were also minimum; under the threshold of 10, quantity of squares increased greatly because that after introduction of threshold, neighbor points of which the gray scale value differences stayed in the threshold scope could be included into the same square, so that other performance indexes such as sum of perimeters also correspondingly increased. When the threshold was further increased to 20 or 30, the quantity of squares again decreased continuously and sum of perimeters also decreased continuously along with it. However, sum of areas continuously increased because, under the increased thresholds, the allowable gray scale value differences of neighbor points increased, so that more pixel points could be included into the same square. In this way, total quantity of squares decreased, the area of a single square increased quickly, and total coverage area of squares in the image also increased correspondingly. Decrease in total quantity of squares led to decrease in edges and sum of perimeters. Building and Flower had relatively low image complexities, indicating that their gray scale distribution was relatively concentrative and simple. Hence, quantity of squares generated from coding continuously decreased with the increase of threshold, and sum of perimeters also decreased continuously with it. However, sum of areas continuously increased. Principles causing such result conform to those in Flight and Peppers. In addition, duration for SNAMG neighbor searching, average duration for perimeter calculation and duration for area calculation of each image was very small as a whole, while their execution efficiency was high. However, they did not vary in certain rules because efficiency of these calculations is not only related to quantity of squares, but is also influenced by size of squares. For example, in case of big square area, side length of the squares is still long even if the total quantity of squares decreases. In this way, durations for neighbor searching and perimeter calculation of a single square will also increase correspondingly.

7. Conclusions

The paper firstly improved the original scanning manner based on the SNAMG representation method in order to adapt it to images with different gray scale texture characteristics; then increased its representation efficiency by virtue of human visual system characteristics, and on this basis put forward algorithms for image neighbor searching, perimeter and area calculation which could be directly executed in gray images, so as to get rid of binary transformation in geometrical property calculation of gray images and increase image processing efficiency. At last, the paper conducted experiments to verify adaptability and high efficiency of these algorithms in images with different complexities. It is shown in experimental results that research in this paper can effectively promote application of SNAMG in image measurement, target recognition, edge detection and other fields.

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, Grant No.KY2015YB291) and Guangxi Scientific Research and Technological Development Project (Grant No. 12239002-8).

References

1. Xing Z, Shui L. (2015) Image Edge Feature Extraction and Refining Based on Genetic-Ant Colony Algorithm. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 13(1), p.p.118-127.
2. J J Laguardia, E Cueto, M Doblare (2005) A Natural Neighbor Galerkin Method with Quadtree Structure. *International Journal for Numerical Methods in Engineering*, 63(6), p.p. 789-812.
3. I Gargantini. (1982) An Effective Way to Represent Quadtrees. *Communications of the ACM*, 25(12), p.p.905-910.
4. V Khanna, P. Gupta, J C Hwang (2003) Maintaining Connected Components in Quadtree-Based Representation of Images. *Iranian Journal of Electrical and Computer Engineering*, 2(1), p.p.53-60.
5. B P Kumar, P Gupta, C J Hwang (2001) An Efficient Quadtree Data Structure for Neighbor Finding Algorithm .*International Journal of Image and Graphics*, 1(4), p.p. 619-633.
6. P Bhattacharya. (2002) *Efficient Neighbor-Finding Algorithm in Quadtree and Octree*. Indian Institute of Technology: Kanpur.

7. S De Bruin, W De Wit, J Van Oort (2004) Using Quadtree Segmentation to Support Error Modeling in Categorical Raster Data. *International Journal of Geographical Information Science*, 18(2), p.p.151-168.
8. C L Wang, S C Wu, Y K Chang (2005) Quadtree and Statistical Model-Based Lossless Binary Image Compression Method. *Imaging Science Journal*, 53(2), p.p. 95-103.
9. G Minglun, Y Yee-Hong (2004) Quadtree-Based Genetic Algorithm and Its Applications to Computer Vision. *Pattern Recognition*, 37(8), p.p.1723-1733.
10. Chuanbo C. (2005) *Study on A Non-Symmetry and Anti-Packing Pattern Representation Method*. Huazhong University of Science and Technology: Wuhan.
11. Peng W, Chuanbo C, Yunping Z. (2009) Image Set-Operation Algorithm based on NAM and Its Experimental Research. *Journal of Yangtze University (Natural Science Edition)*, 6(2), p.p.76-79.
12. Weiju H. (2009) *Study on Multiple Sub-Pattern Image Representation and Retrieve Methods based on NAM*. Huazhong University of Science and Technology: Wuhan.
13. Dehong Q, Xinxin P, Chuanbo C, Shaohong F. (2008) Sequence Classification based on Feature Subsequence Optimized through Semi definite Program. *Journal of Chinese Computer Systems*, 29(11), p.p.2083-2086.
14. Junjie P. (2011) *Graph-based NAM Representation and Salient Region Detection*. Huazhong University of Science and Technology: Wuhan.
15. Jie H, Hui G. Jie He, Yunping Zheng, Hui Guo (2012) A Novel Gray Image Representation Method Based on NAM Using Non-overlapping Square Subpatterns. *Applied Mechanics and Materials*, 133-134, p.p.760-764.
16. Yunping Z, Chuanbo C. (2008) An Optimization Strategy of NAM using Raster Scanning. *Journal of Huazhong University of Science and Technology (Natural Science Edition)*, 36(8), p.p.1-4.
17. Yong L. (2011) *A Unified Objective and Subjective Image Quality Evaluation Model based on The Human Visual System*. Jilin University: Changchun.
18. B P Kumar, P Gupta, C J Hwang. (2001) An Efficient Quadtree Data Structure for Neighbor Finding Algorithm. *International Journal of Image and Graphics*, 1(4), p.p.619-633.