IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 5, Issue 1, Ver. I (Jan - Feb. 2015), PP 54-59
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Finding Neighbors in Images Represented By Quadtree

Hassan Id Ben Idder[1], Nabil Laachfoubi[2]
[1, 2](Department of Mathematics, IR2M laboratory,Faculty of Science and Technology/ University Hassan 1st, 26000, Settat, Morocco)

Abstract: *In this paper, we propose an algorithm for neighbors finding in images represented by quadtree data structure. We first present a scheme for addressing image pixels that takes into account the order defined on a quadtree blocks. We study various properties of this method, and then we develop a formula that establishes links between quadtree blocks. Based on these results, we present an efficient algorithm that generates the list of location codes of all possible neighbors, in all directions, for a given block of the quadtree.*

Keywords: *Image processing, quadtree, neighbors finding, pixel ordering.*

I. Introduction

Neighbors finding is a basic operation for the majority of tasks to be performed on an image. It is simple, using matrix representation through addressing indices (row, column), but it is complex to achieve using quadtree representation. This complexity is due to the fact that a point on the boundary of a block has neighbors in the same block, has neighbors in the top level block or has neighbors in lower level block. In most cases, a block can have more than one neighbor in a given direction; the problem is then to find its neighboring blocks, in all directions, all relying solely on the quadtree structure of the image. This problem has been the subject of several research [1]–[4] since the introduction of quadtree by Finkel and Bentley[5] in the seventies. Among the most important algorithms we mention: the algorithm proposed by Samet [6], it finds the neighbor of equal or larger size in a given direction. This algorithm is based on the pointer based implementation of the quadtree. Gargantini [7] developed an algorithm for finding neighbors in a linear representation of the quadtree; the algorithm finds only the neighbors having the same size as the starting block. The Schrack's algorithm [3] also allows finding the neighbors of the same size in a linear quadtree. This algorithm computes in a constant time, the location codes of the neighboring blocks without determining their existence in the quadtree structure. Recent work of Aizawa [8], [9] presents an algorithm that finds neighbors of equal or larger sizes in a constant time, this algorithm is based on a modified linear representation of the quadtree that stores the level differences between adjacent blocks, the algorithm uses complex data structure that requires more memory space.

In this paper, we present, in the first instance, a new scheme for addressing the image pixels which allows rearranging the pixels according to their position in the quadtree. We follow up by introducing a formula that establishes the link between the location codes of any two quadtree blocks. This formula is then used to develop an algorithm for finding the neighbors, in all directions, of a given block. The algorithm does not verify the existence of neighbors in the underlying quadtree structure.

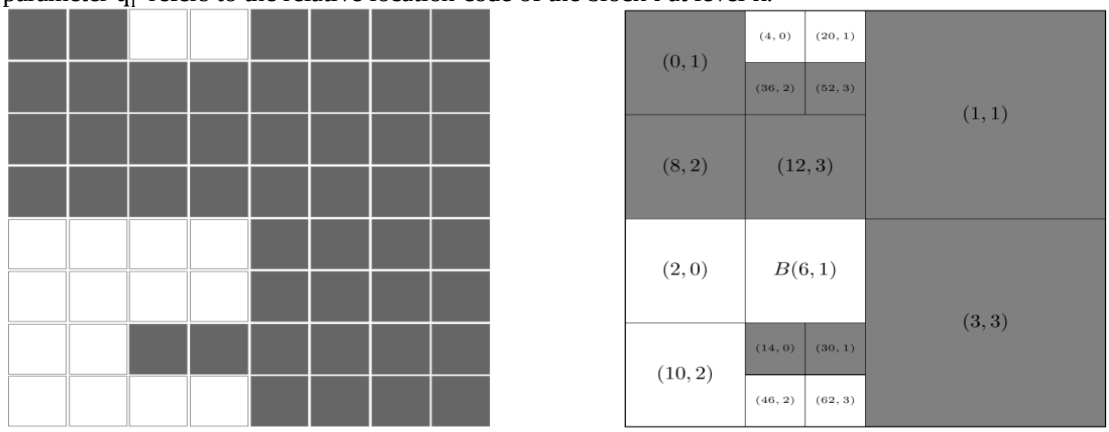
II. Notations andDefinitions

A. Absolute and relative location codes

In the remainder of this document, the notation $(\cdots)_2$ refers to the binary representation of an integer, the operator $\ll$ is used to perform a left shift of bits, and the AND operator denotes the logical "and" between two expressions. Given a binary image of size $2^n \times 2^n$ and its quadtree decomposition (Fig. 1), it is assumed that the smallest division represents the level n of the quadtree. The four blocks resulting from a decomposition of the image are ordered using both a relative location code and an absolute location code. The relative location code, denoted $q_{i=0,\cdots,3}$ is defined as follows: the value 0 represents the northwest block (0), 1 for the northeast block, 2 for the southwest block and 3 for the southeast block. It locates a block with respect to its three neighbors, belongings to the same decomposition (Fig. 2). the absolute location code, denoted $c_{i=0,\cdots,3} \in \{0, \cdots, 2^{2n} - 1\}$, It is constructed by putting side by side the binary value of the relative location of the quadrant subdivision from which it originates (Fig. 3). Formally, the absolute location code is defined by:

$$c_i = ((q_i^{n-1} q_i^{n-1})_2 \cdots (q_i^0 q_i^0)_2)_2 = \sum_{k=0}^{n-1} q_i^k 2^{2k} \qquad (1)$$

The parameter q_i^k refers to the relative location code of the block i at level k.



(a) **(b)**

Figure 1:(a) An image of size 8×8 and (b) its quadtree decomposition. Each block is labeled by the notation(c, q) , which corresponds to the block with the absolute location code c and the relative location code q.

The absolute location code can also be defined as the interleaving of the bits that comprise the values of the pixel's x and y coordinate in upper left corner of the block [7]. Given a pixel with coordinates $x = (x_0 x_1 \cdots x_{n-1})_2$ and $y = (y_0 y_1 \cdots y_{n-1})_2$, the corresponding absolute location code has the following binary structure:

$$c = (x_0 y_0 x_1 y_1 \cdots x_{n-2} y_{n-2} x_{n-1} y_{n-1})_2 \quad (2)$$

The absolute location code used in this document is equivalent to the location codes assigned by Gargantini [7] to linear quadtree elements, except that: the two most significant bits represent the relative location code of the block at level n (the smallest division in the quadtree), while the two least significant bits represent the relative location code of the block at level 0 (quadtree root). This difference has a big impact on how the image pixels are ordered. Thus, by sorting image pixels according to their absolute location code, we get a new addressing scheme which respects the hierarchical appearance of quadtree.

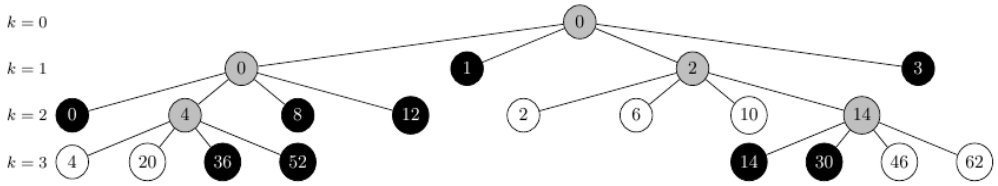


Figure 2: Quadtree of the image in (Fig. 1). Each node is labeled with its absolute location code, The black elements (respectively, the white elements) represent image blocks containing only the pixels belonging to the object (respectively, to the background), while the gray elements represent blocks containing both object pixels and background pixels.

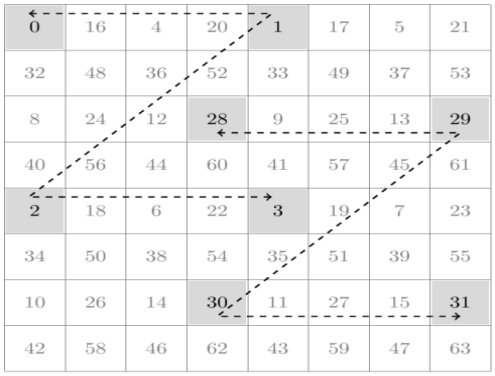


Figure 3: The scan order produced by the absolute location codes for addressing the pixels of an image of size

8x8. The four successive elements of the form {4a, 4a+1, 4a+2, 4a+3} form a pattern of Z.

In the remainder of this paper, a quadtree block is referenced by the notation (c, q, k) , which means: the block with the absolute location code c and the relative location code q on the level k of the quadtree. When there is no ambiguity, the index k is omitted and only the notation (c, q) is used to denote a quadtree block.

Property 1: Given a quadtree block, with an absolute location code c. The relative location code q associated with that block at level k is given by:

$$q = (11)_2 \text{ AND } (c \ll 2k) \tag{3}$$

Property 2: Given two adjacent quadtree blocks (c_i, q_i) and (c_j, q_j) belonging to the same parent block at level k. Their absolute location codes are linked by the following formula:

$$c_i = c_j + (q_i - q_j)2^{2k} \ ; \ (i, j) \in \{0, \cdots, 3\} \tag{4}$$

Proposition 1 (Generalization of property 2 to arbitrary blocks): Let k_0 be the level of the smallest subdivision which contains both the block (c_i, q_i, k_i) and the block(c_j, q_j, k_j). The absolute location codes c_i and c_j are linked by the following formula:

$$c_j = c_i + \sum_{k=k_0}^{\max(k_i, k_j)} (q_j^k - q_i^k)2^{2k} \tag{5}$$

Proof:

If c_i and c_j belongs to the same parent block, then $k_0 = k_i = k_j$, we are left with the formula (4), otherwise we compute the absolute location codes c_i' and c_j' of the parent blocks of c_i andc_j using formula (4), hence: $c_i' = c_i + q_i^k 2^{2(k-1)}$and $c_j' = c_j + q_j^k 2^{2(k-1)}$. If c_i' and c_j' belongs to the same parent block, then the formula (4) gives: $c_j' = c_i' + (q_j^{k-1} - q_i^{k-1})2^{2(k-1)}$ and by combining the above three equations we obtain:

$$c_j = c_i + (q_j^k - q_i^k)2^{2k} + (q_j^{k-1} - q_i^{k-1})2^{2(k-1)}$$

If c_i' and c_j' belong to two different parent blocks, we repeat recursively the same procedure for the parent blocks of c_i' and c_j'. The procedure stops when the two blocks belong to the same parent block.

B. Neighbors in quadtree

Samet [6] defined the neighbor of a quadtree block B in a direction D as the smallest block B' (possibly gray) adjacent to the block B in the direction D and having a size greater than or equal to the size of block B. The direction can be a direction of direct neighborhood (north, south, east and west) or a direction of indirect neighborhood (northeast, northwest, southeast, and southwest) (Fig. 4). Several algorithms that have been proposed subsequently to find neighbors in a quadtree are based on this definition. In this paper we also adopt the same definition to characterize the neighboring blocks in a quadtree.

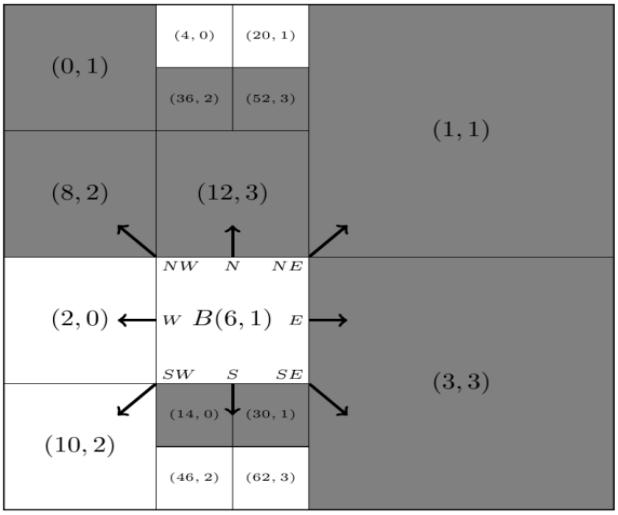


Figure 4 : Directions of direct neighborhood (N, E, S and W) and indirect neighborhood (NW, NE, SE and

SW) for the block $B(6,1,2)$

III. Algorithms

Formula (5) establishes the link between any two blocks of the quadtree based on their absolute and relative location codes. However, it does not directly find neighbors of a given block. In this section we propose an algorithm, which uses the results of the previous section, to generate the list of neighbors of a given block in all directions. We proceed in two steps: first, we describe an algorithm that finds neighbors of equal size, and then we follow up with a second algorithm for findings neighbors of larger sizes. Combining the two algorithms we obtain an algorithm that generates the absolute location codes of all possible neighbors in all directions.

A. Neighbors of equal size

Let $B(c,q,k)$ be a quadtree bloc, the relative location codes of its two neighbors of equal size in the west and east directions (respectively south and north directions) are defined byq XOR 1 (respectivelyq XOR 2). This allows us to simplify the formula 6 and then reduce the number of parameters to be defined:

$$c_i = c + \sum_{l=k_0}^{k} \left(q^l \text{ XOR } d - q^l\right) 2^{2l} \qquad (6)$$

The parameter d depends on the direction of the search; it takes the value 1 for the west and east directions and the value 2 for the south and north directions. k_0 represents the level of the small subdivision that contains both the block and its neighbor (i.e. k_0 is the level of the lowest common parent block of the starting block and its neighbor), its value is determined by analyzing the position of the block with respect to the border of his parent block: if a block is located in the northwest or southwest (respectively northeast or southeast), then its neighbor in the east (respectively the west) also belongs to the same subdivision as the starting block. We put $k_0 = k - 1$, if the block and its neighbor does not belong to the same parent block in the level k_0, we decrease the value of k_0 and we proceed in the same way for the parent of the block and that of his neighbor. The procedure stops when both parent blocks belong to the same subdivision. Note that the neighbors of a block $B(c,q)$ in the northwest and northeast directions (respectively in the southwest and southeast directions) are also the neighbors in the east and west directions of the block $B'(c',q')$, itself a neighbor of the block $B(c,q)$ in the north direction (respectively in the south direction). So the problem of finding neighbors in the directions of indirect neighborhood (northwest, northeast, southwest and southeast) is reduced to a problem of finding the neighbors in the directions of direct neighborhood (north, south, east and west).

Let $k_0 = k - 1$, Q1 = {0,2} and Q2 = {1,3} (respectively : Q1 = {0,1} et Q2 = {2,3})
If $q^k \in Q1$ then while $q^{k_0} \in Q1$ and $k_0 > 0$ decrease the value of k_0
Otherwise, if $q^k \in Q2$ then while $q^{k_0} \in Q2$ and $k_0 > 0$ decrease the value of k_0
Let $c' = c$ and $c'' = c + (q^k \text{ XOR } 1 - q^k)2^{2k}$.
Compute c' using formulas 4 and 7.
If $q^k \in Q1$ then :
The block $(c'q,k)$ is the neighbor of (c,q,k) in the west direction (respectively in the south direction).
The block (c'',q,k) is the neighbor of (c,q,k) in the east direction (respectively in the north direction).
If $q^k \in Q2$ switch the roles of $(c'q,k)$and (c'',q,k).

Algorithm 1: Finding neighbors of equal size in the north, south, east and west directions.

The algorithm finds the four neighbors of a given block B in the directions of direct neighborhood (north, south, east and west). To find the remaining four neighboring blocks in the directions of indirect neighborhood (northwest, northeast, southwest and southeast), we apply the same algorithm to two neighbors B' and B'' of the block B. The neighbors B' and B'' are either in the north and the south directions respectively or in the east and west directions respectively. In the first case the four neighbors of the block B in the directions of indirect neighborhood are the neighbors of B' and B'' in the east and west directions, whereas in the second case the four neighbors of the block B in the directions of indirect neighborhood are the neighbors of B' and B'' in the north and south directions (Fig. 5).

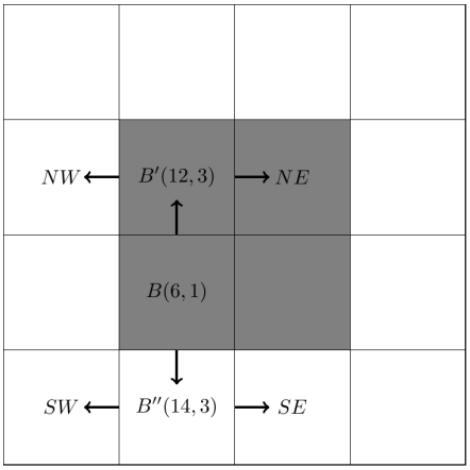


Figure 5: Finding neighboring blocks in the directions of indirect neighborhood (northwest, northeast, southwest and southeast), the four neighbors of the block B in the directions of indirect neighborhood are the neighbors of B' and B'' in the east and west directions

B. Neighbors of large size

The neighbors of large size of a block $B(c, q, k)$ are the parent blocks of its eight neighbors of equal size (neighbors generated by algorithm 1), that are not a parent block of the block $B(c, q, k)$ The search is performed recursively for the eight neighbors of the block $B(c, q, k)$, it stops when a parent block of the block $B(c, q, k)$ is met.

Let V be the list of neighbors generated by algorithm 1
For each block $B(c, q, k) \in V$
Set $k' = k - 1$ and $c' = c$;
Compute the parent: $c'' = c' + q2^{2k'}$
If the block (c'', q, k') is not a parent of (c, q, k) then (c'', q, k') is a neighbor of $B(c, q, k)$, Let then c' = c'' and k' = k'-1 and restart from step 2.b
Fin.

Algorithm 2: Finding neighbors of larger sizes in all directions.

To illustrate how those two algorithms work, we will compute the location codes of the neighbors of the block $B(6,1,2)$ shown in (Fig. 4). The results are described below (Table 1).

Table 1 : Neighbors of block $B(6,1,2)$ using algorithms 1 and 2

Block	Direction	Neighbors	
		Equal size	Greater size
B(6,1,2)	North	(12,3,2)	(0,1,1)
	Northeast	(9,2,2)	(1,1,1)
	East	(3,0,2)	(3,1,1)
	Southeast	(11,2,2)	(3,1,1)
	South	(14,3,2)	-
	Southwest	(10,2,2)	-
	West	(2,0,2)	-
	Northwest	(8,2,2)	(0,1,1)

Algorithms 1 and 2 have a run time which is proportional to the difference between the level k of a block and the level k' of its neighbors in the quadtree $O(|k - k'| \leq n)$. The algorithm 2 is recursive, it stops when it reaches a quadtree leaf (block located at level $(n - 1)$. Furthermore, note that these two algorithms are independent of the way the quadtree is represented in memory. In fact, if the blocks are labeled by their absolute location codes, these algorithms can be applied to both a linear quadtree and a pointer based quadtree.

IV. Conclusion

In this article we introduced a new scheme for addressing image pixels that rearrange the points according to their position in the quadtree. This method allowed us to develop an algorithm for finding

neighbors of a given block in the quadtree. The proposed algorithm generates a list of location codes of all possible neighbors without checking their existence in the quadtree structure, additional verification may be performed to determine whether the neighbors exist or not in the underlying quadtree data structure.

References

[1] H. Samet, "Neighbor finding in images represented by octrees," Comput. Vision, Graph. Image Process., vol. 46, no. 3, pp. 367–386, 1989.

[2] C.-Y. Huang and K.-L. Chung, "Faster neighbor finding on images represented by bincodes," Pattern Recognit., vol. 29, no. 9, pp. 1507–1518, 1996.

[3] G. Schrack, "Finding neighbors of equal size in linear quadtrees and octrees in constant time," CVGIP Image Underst., vol. 55, no. 3, pp. 221–230, 1992.

[4] J. Vörös, "A strategy for repetitive neighbor finding in images represented by quadtrees," Pattern Recognit. Lett., vol. 18, no. 10, pp. 955–962, 1997.

[5] R. A. Finkel and J. L. Bentley, "Quad trees a data structure for retrieval on composite keys," Acta Inform., vol. 4, no. 1, pp. 1–9, 1974.

[6] H. Samet, "Neighbor finding techniques for images represented by quadtrees," Comput. Graph. Image Process., vol. 18, no. 1, pp. 37–57, 1982.

[7] I. Gargantini, "An Effective Way to Represent Quadtrees," Commun. ACM, vol. 25, no. 12, pp. 905–910, Dec. 1982.

[8] K. Aizawa, K. Motomura, S. Kimura, R. Kadowaki, and J. Fan, "Constant time neighbor finding in quadtrees: An experimental result," in Communications, Control and Signal Processing, 2008. ISCCSP 2008. 3rd International Symposium on, 2008, pp. 505–510.

[9] K. Aizawa and S. Tanaka, "A Constant-Time Algorithm for Finding Neighbors in Quadtrees," Pattern Anal. Mach. Intell. IEEE Trans., vol. 31, no. 7, pp. 1178–1183, Jul. 2009.