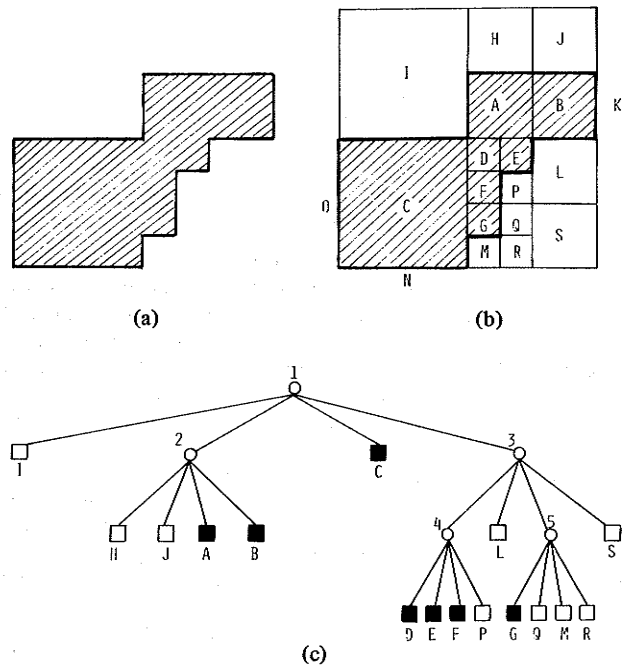

Fig. 1. An image, its maximal blocks, and the corresponding quadtree. Blocks in the image are shaded. (a) Sample image. (b) Block decomposition of the image in (a). (c) Quadtree representation of the blocks in (b).

Computing Perimeters of Regions in Images Represented by Quadtrees

HANAN SAMET

Abstract—**An algorithm is presented for computing the total perimeter of a region in a binary image represented by a quadtree. The algorithm explores each segment of the boundary of the region once and only once. Analysis of the algorithm shows that its average execution time is proportional to the number of leaf nodes in the quadtree.**

Index Terms—**Image processing, perimeter, quadtrees, region representation.**

I. Introduction

Perimeter computation is a basic operation in image processing [10]. The standard algorithms use either an array or a chain code representation [2] for a region in a two-valued ("binary") image. In this correspondence we present an algorithm for computing the total perimeter (i.e., the length of the chain codes corresponding to the boundary of the region) of a region in a binary image that is represented by a quadtree [1], [3]-[8], [11].

We assume that the given image is a $2^n \times 2^n$ array of unit square "pixels." The quadtree is an approach to image representation based on successive subdivision of the array into quadrants. In essence, we repeatedly subdivide the array into quadrants, subquadrants, ... until we obtain blocks (possibly single pixels) which consist entirely of either 1's or 0's. This process is represented by a tree of out-degree 4 in which the root node represents the entire array, the four sons of the root node represent the quadrants, and the terminal nodes correspond to those blocks of the array for which no further subdivision is necessary. For example, Fig. 1(b) is a block decomposition of the region in Fig. 1(a), while Fig. 1(c) is the corresponding quadtree. In general, BLACK and WHITE square nodes represent terminal nodes consisting entirely of 1's and 0's, respectively. Circular nodes, also termed GRAY nodes, denote nonterminal nodes.

Sections II-V present and analyze the algorithm. Included is an informal description of the algorithm along with motivating considerations. The actual algorithm is given using a variant of Algol 60 [9].

II. Definitions and Notation

Let each node in a quadtree be stored as a record containing six fields. The first five fields contain pointers to the node's father and its four sons, i.e., quadrants, labeled *NW*, *NE*, *SE*, and *SW*. Given a node P and a son I, these fields are referenced as FATHER(P) and SON(P, I), respectively. We assume that the FATHER of the root of the tree is NULL. We can determine the specific quadrant in which a node, say P, lies relative to its father by use of the function SONTYPE(P), which has a value of I if SON(FATHER(P), I) = P. The sixth field, named NODETYPE, describes the contents of the block of the image which the node represents, i.e., WHITE if the block contains no 1's, BLACK if the block contains only 1's, and GRAY if it contains 0's and 1's.

Manuscript received May 7, 1979; revised February 2, 1981. This work was supported by the Defense Advanced Research Projects Agency and the U.S. Army Night Vision Laboratory under Contract DAAG-53-76C-0138 (DARPA Order 3206).

The author is with the Department of Computer Science, University of Maryland, College Park, MD 20742.

0162-8828/81/1100-0683$00.75

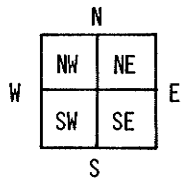

Fig. 2. Relationship between a block's four quadrants and its boundaries.

Let the four sides of a node's block be called its N, E, S, and W sides. They are also termed its boundaries and at times we speak of them as if they are directions. Fig. 2 shows the relationship between the quadrants of a node's block and its boundaries. A node, say Q, is said to be a *neighbor* of another node, say P, in direction D if Q corresponds to the smallest block having a side that is adjacent to all of side D of P's block (not just at a corner; also, a node corresponding to a bigger block cannot be returned when an equal adjacent one exists). For example, in Fig. 1, BLACK node A is the neighbor of BLACK node E in the northern direction; similarly, GRAY node 4 is the southern neighbor of node A.

The algorithm for the computation of the perimeter makes use of the spatial relationships between the various sides. This is accomplished by use of the functions OPSIDE, CSIDE, and CCSIDE. OPSIDE(B) is a side facing side B; e.g., OPSIDE(E) = W. CSIDE(B) and CCSIDE(B) correspond to the sides adjacent to side B in the clockwise and counterclockwise directions respectively, e.g., CSIDE(E) = S and CCSIDE(E) = N.

In order to determine a node's neighbor in a specified direction, we have to traverse ancestor links until a common ancestor of the two nodes is found. Once the common ancestor is located, we descend along a path that retraces the previous path with the modification that each step is a reflection of the corresponding step about the axis formed by the common boundary between the two nodes. For example, when attempting to locate the northern neighbor of nodes E (i.e., node A) in Fig. 1, node 1 is the common ancestor, and the northern edge of the block corresponding to node E is the common boundary. During this process we pass through nodes 4, 3, 1, 2, and A in order. The expression of this procedure is facilitated by the following predicates and functions. ADJ(B, I) is true if and only if quadrant I is adjacent to boundary B of the node's block, e.g., ADJ(N, NW) is true. REFLECT(B, I) yields the SONTYPE value of the block of equal size that is adjacent to side B of a block having SONTYPE value I, e.g., REFLECT(W, NW) = NE, REFLECT(E, NW) = NE, REFLECT(N, NW) = SW, and REFLECT(S, NW) = SW. QUAD(B, C) is the quadrant which is bounded by boundaries B and C (if B and C are not adjacent boundaries, then the value of QUAD(B, C) is undefined); e.g., QUAD(N, W) = NW.

Given a quadtree corresponding to a $2^n \times 2^n$ array, we say that the root node is at level n, and that a node at level i is at a distance of $n - i$ from the root of the tree. In other words, for a node at level i, we must ascend $n - i$ FATHER links to reach the root of the tree. Note that the farthest node from the root of the tree is at level ≥ 0. A node at level 0 corresponds to a single pixel in the image. Also, we say that a node is of size 2^S if it is found at level S in the tree, i.e., it has a side length of 2^S.

III. Algorithm

The perimeter computation algorithm traverses the quadtree in postorder (i.e., the sons of a node are visited first). Each segment of the boundary of the region is visited once and only once. For each BLACK terminal node, say P, visit all of the nodes whose corresponding blocks have a boundary in common with P's block. These are termed P's northern, eastern, southern, and western adjacencies. For each of the visited nodes that is WHITE, the length of the common boundary is included in the value of the perimeter. For example, given BLACK node A in Fig. 1(b), nodes I, H, B, E, and D are visited. Of these nodes, only I and H are WHITE and thus the only contribution to the value of the perimeter is the length of the boundary segments between node A and I and A and H (denoted by AI and AH, respectively).

The main procedure is termed PERIMETER and is invoked with a pointer to the root of the quadtree representing the region and an integer corresponding to the resolution of the image (e.g., n for a $2^n \times 2^n$ image array). PERIMETER traverses the tree and controls the exploration of the adjacencies of each BLACK node. For each BLACK node, say P, GTEQUAL_ADJ_NEIGHBOR locates a neighboring node, say Q, of greater or equal size along a specified direction, say D. If Q is WHITE, then the contribution to the perimeter is the size of P. If Q is BLACK, then no contribution is made to the perimeter. If the boundary of P in direction D is on the border of the image, then no neighbor exists in the specified direction and NULL is returned. In such a case the contribution to the perimeter of the boundary of P in direction D is equal to the size of P. For example, in Fig. 1(b), the neighbors of node C in the western and southern directions are NULL and contribute the length of the sides of C that are adjacent to the border of the image, i.e., CO and CN, respectively. If the boundary of P in direction D is not on the border of the image, and no neighboring BLACK or WHITE node exists satisfying our size criteria, then a pointer to a GRAY node of equal size is returned [e.g., the eastern neighbor of node C in Fig. 1(b)]. In such a case procedure SUM_ADJACENT continues the search by examining all nodes of smaller size that are adjacent to P's boundary in direction D and accumulating the sizes of all such nodes that are WHITE [e.g., block M for the eastern border of node C in Fig. 1(b)].

An alternative method of computing the perimeter is to apply the algorithm in [1] which converts a quadtree representation to a four direction chain code and then simply sum the lengths of the segments.[1] The algorithm of this correspondence is simpler since it does not require the segments to be traversed in sequence around the boundary of the region. We need only ensure that each boundary segment is visited once and only once. This is clearly true since during the tree traversal, the adjacencies of each BLACK node are explored at least once; on the other hand, each boundary segment is only explored once since it must adjoin a WHITE node and our algorithm does not explore adjacencies of WHITE nodes.

As an example of the application of the algorithm, consider the region given in Fig. 1(a). Fig. 1(b) is the corresponding block decomposition and Fig. 1(c) is its quadtree representation. All of the BLACK nodes have labels ranging between A and G, while the WHITE nodes have labels ranging between H and S. The BLACK nodes are labeled in the order in which their adjacencies are explored by PERIMETER. WHITE nodes H–Q are labeled in the order in which they are first visited by the combination of GTEQUAL_ADJ_NEIGHBOR and SUM_ADJACENT. Thus, the adjacencies of node A have been explored before those of nodes B, C, etc. The value of the perimeter is obtained by visiting the boundary segments in the order AH, AI, BJ, BK, BL, CI, CM, CN, CO, EL, EP, FP, GQ, and GM. Assuming $n = 3$ (i.e., blocks D, E, F, G, M, P, Q, and R are single pixels), the perimeter is 28. Note that nodes D, R, and S do not contribute to the value of the perimeter since none of their sides adjoin the boundary of the region. Table I contains a node-by-node trace of the perimeter computation process.

The actual algorithm is given below using a variant of Algol.

[1] If we were attempting to estimate the perimeter of the original object from the lengths of the border of the digital object, a higher order chain code such as eight-neighbor might be more accurate, but our aim in this correspondence is simply to measure the border of the digital object itself which is by definition a set of horizontal and vertical "cracks."

```
integer procedure PERIMETER(P, LEVEL);
/* Find the perimeter of a region represented by a quadtree
   rooted at node P that spans a 2↑LEVEL by 2↑LEVEL space */
begin
   value node P;
   node Q;
   value integer LEVEL;
   integer LEN;
   quadrant I;
   direction D;
   LEN←0;
   if GRAY(P) then
      begin
         for I in {"NW," "NE," "SW," "SE"} do
               LEN←LEN+PERIMETER(SON(P, I), LEVEL-1);
      end
   else if BLACK(P) then
      begin
         for D in {"N," "E," "S," "W"} do
            begin
               Q←GTEQUAL_ADJ_NEIGHBOR(P, D);
               LEN←LEN + if NULL(Q) or WHITE(Q) then 2↑LEVEL
                         else if GRAY(Q) then
                            SUM_ADJACENT(Q, QUAD(OPSIDE(D), CSIDE(D)),
                                            QUAD(OPSIDE(D), CCSIDE(D)),
                                            LEVEL)
                         else 0;
            end;
      end;
   return (LEN);
end;
node procedure GTEQUAL_ADJ_NEIGHBOR(P, D);
/* Return the neighbor of node P in horizontal or vertical di-
   rection D which is greater than or equal in size to P. If such
   a node does not exist, then a GRAY node of equal size is
   returned. If this is also impossible, then the node is adja-
   cent to the border of the image and NULL is returned */
begin
   value node P;
   node Q;
   value direction D;
   if not NULL(FATHER(P)) and ADJ(D, SONTYPE(P)) then
      /* Find a common ancestor */
      Q←GTEQUAL_ADJ_NEIGHBOR(FATHER(P), D)
   else Q←FATHER(P);
/* Follow the reflected path to locate the neighbor */
return (if not NULL(Q) and GRAY(Q) then SON(Q, REFLECT(D, SONTYPE(P)))
        else Q);
end;
integer procedure SUM_ADJACENT(P, Q1, Q2, LEVEL);
/* Find all WHITE leaves in quadrants Q1 and Q2 of the sub-
   quadtree rooted at node P of size 2↑LEVEL */
begin
   value node P;
   value quadrant Q1, Q2;
   value integer LEVEL;
   return (if GRAY(P) then SUM_ADJACENT(SON(P, Q1), Q1, Q2, LEVEL-1)
                         + SUM_ADJACENT(SON(P, Q2), Q1, Q2, LEVEL-1)
           else if WHITE(P) then 2↑LEVEL
           else 0);
end;
```

IV. Analysis

The running time of the perimeter computation algorithm, measured by the number of nodes visited, depends on the time spent locating adjacent WHITE nodes and on the size of the quadtree. Adjacent WHITE nodes are located by procedures GTEQUAL_ADJ_NEIGHBOR and SUM_ADJACENT. They are invoked four times for each BLACK node. The amount of work performed by these procedures is obtained by considering the number of nodes that are visited when an adjacency is being explored. Recall that we must find the neighbor, and if it is GRAY, then visit all adjacent WHITE nodes of smaller size. In the worst case we are at level $n-1$, with a GRAY neighbor, and all adjacent nodes are at level 0. In such a case we must visit 2^n nodes. For example, consider Fig. 1 where

TABLE I
TRACE OF THE PERIMETER COMPUTATION FOR FIG. 1

node	side	neighbor	segment	Contribution to Perimeter	Perimeter
1					
I				ø	ø
2					
H				ø	ø
J				ø	ø
A	N	H	AH	2	2
A	E	B	AB	ø	2
A	S	4			
A	S	D	AD	ø	2
A	S	E	AE	ø	2
A	W	I	AI	2	4
B	N	J	BJ	2	6
B	E	K	BK	2	8
B	S	L	BL	2	10
B	W	A	AB	ø	10
C	N	I	CI	4	14
C	E	3			
C	E	5			
C	E	M	CM	1	15
C	E	G	CG	ø	15
C	E	4			
C	E	F		ø	15
C	E	D		ø	15
C	S	N	CN	4	19
C	W	O	CO	4	23
3					
4					
D	N	A	AD	ø	23
D	E	E	DE	ø	23
D	S	F	DF	ø	23
D	W	C	CD	ø	23
E	N	A	AE	ø	23
E	E	L	EL	1	24
E	S	P	EP	1	25
E	W	D	DE	ø	25
F	N	D	DF	ø	25
F	E	P	FP	1	26
F	S	G	FG	ø	26
F	W	C	CF	ø	26
P				ø	26
L				ø	26
5					
G	N	F	FG	ø	26
G	E	Q	GQ	1	27
G	S	M	GM	1	28
G	W	C	CG	ø	28
Q				ø	28
M				ø	28
R				ø	28
S				ø	28

$n = 3$ and we wish to visit the nodes adjacent to the node labeled C (i.e., nodes D, F, G, and M). We must visit the root of the quadtree as well as C's neighboring GRAY node 3 and all of its NW and SW sons, i.e., a complete binary tree of height 2. In total, $2^3 = 8$ nodes are visited (nodes 1, 3, 4, D, F, 5, G, and M).

Our analysis assumes a $2^n \times 2^n$ random image in the sense that a node is equally likely to appear in any position and level in the quadtree. This means that we assume that all configurations of adjacent nodes of varying sizes have equal probability. This is different from the more conventional notion of a random image which implies that every block at level 0 (i.e., pixel) has an equal probability of being BLACK or WHITE. Such an assumption would lead to a very low probability of any node corresponding to blocks of size larger than 1. Clearly, for such an image the quadtree is the wrong representation.

Theorem 1: The average of the maximum number of nodes visited by each invocation of GTEQUAL_ADJ_NEIGHBOR and SUM_ADJACENT is 5.

Proof: Given a node P at level i and a direction S, there are $2^{n-i}(2^{n-i} - 1)$ neighbor pairs. $2^{n-i} \cdot 2^0$ have their nearest common ancestor at level n, $2^{n-i} \cdot 2^1$ at level $n - 1, \cdots$ and $2^{n-i} \cdot 2^{n-i-1}$ at level $i + 1$. For each node at level i having a common ancestor at level j, the maximum number of nodes that will be visited by GTEQUAL_ADJ_NEIGHBOR and SUM_ADJACENT is

$$(j - i) + (j - i - 1) + \sum_{k=0}^{i} 2^k = 2(j - i - 1) + 2^{i+1}.$$

This is obtained by observing that the common ancestor is at a distance of $j - i$ and that a node at level i has a maximum of 2^i adjacent nodes (all appearing at level 0). Assuming that node P is equally likely to occur at any level i and at any of the $2^{n-i}(2^{n-i} - 1)$ positions at level i, then the average of the maximum number of nodes visited by GTEQUAL_ADJ_NEIGHBOR and SUM_ADJACENT is

$$\frac{\sum_{i=0}^{n-1} \sum_{j=i+1}^{n} 2^{n-i} \cdot 2^{n-j}(2(j - i - 1) + 2^{i+1})}{\sum_{i=0}^{n} 2^{n-i}(2^{n-i} - 1)}. \quad (1)$$

Making use of the following identities in the numerator of (1) leads to (2):

$$\sum_{j=0}^{n} \frac{1}{2^j} = 2 \cdot \left(1 - \frac{1}{2^{n+1}}\right)$$

$$\sum_{j=0}^{n} \frac{j}{2^j} = 2 - \frac{n+2}{2^n}$$

$$\frac{20}{3} \cdot 2^{2n} - (3n + 2) \cdot 2^{n+1} - \frac{8}{3}. \quad (2)$$

The denominator of (1) can be manipulated in a similar manner to yield

$$\frac{1}{3} \cdot (2^{2n+2} - 3 \cdot 2^{n+1} + 2). \quad (3)$$

Substituting (2) and (3) into (1) results in

$$5 - \frac{3 \cdot (3n + 7) \cdot 2^{n+1} + 18}{2^{2n+2} - 3 \cdot 2^{n+1} + 2}$$

≈ 5 as n gets large

$\leqslant 5$. Q.E.D.

Lemma 1: The number of nodes in a quadtree having B and W BLACK and WHITE, respectively, nodes is bounded by $4/3 \cdot (B + W)$.

Proof: Let G denote the number of nonterminal nodes. Given G nonterminal nodes and $B + W$ terminal nodes, we have $G + B + W - 1$ edges (since the tree is an acyclic graph). Counting another way, by the number of sons, we find that there are $4G$ edges. Thus, $4G = G + B + W - 1$ or $G = (B + W - 1)/3$. Therefore, the total number of nodes in the quadtree (i.e., $G + B + W$) is $(B + W - 1)/3 + B + W = (4 \cdot (B + W) - 1)/3$ and our result follows. Q.E.D.

We can now prove the following.

Theorem 2: The average execution time of the perimeter computation algorithm is of order $B + W$.

Proof: From Theorem 1 we have that for each adjacency involving a BLACK node, GTEQUAL_ADJ_NEIGHBOR and SUM_ADJACENT result in an average maximum of 5 nodes being visited. There are four adjacencies for each BLACK node. Thus, these two procedures contribute $4B \cdot 5$. From Lemma 1 we have that the number of nodes in the quadtree is bounded by $4/3 \cdot (B + W)$. This quantity correlates with the work performed by procedure PERIMETER since each node in the quadtree is visited by the traversal. Summing up these values we have $4B \cdot 5 + 4/3 \cdot (B + W) = 4/3 \cdot (16 \cdot B + W)$. Q.E.D.

V. CONCLUDING REMARKS

An algorithm has been presented for computing the total perimeter of a region in a binary image represented by a quadtree. The algorithm has been shown to have an average execution time proportional to the number of blocks comprising the region and its background. Note that the number

Fig. 3. Altered image for the alternative perimeter computation method.

of BLACK blocks (i.e., the image complexity) dominates the execution time of the algorithm. It should be clear that if the image contains more than one region, then the algorithm will return the total perimeter of all the regions. Similarly, if holes are present, their boundaries are also included in the value of the perimeter obtained by this algorithm. Note that if we first labeled the connected components (i.e., regions) of the image [12], then the perimeter of each region could be separately computed.

Our algorithm examines the four adjacencies of each BLACK node (i.e., in the N, E, S, and W directions). An alternative is to only examine for each BLACK node the adjacent WHITE nodes in the E and S directions. The difficulty with such an approach is that it ignores the contributions of the N and W segments of the region's boundary (e.g., CO, CI, AI, AH, BJ in Fig. 1). This can be overcome by surrounding the image by six WHITE blocks, as shown in Fig. 3, and for each BLACK node examine the adjacent WHITE nodes in the E and S directions and for each WHITE node examine the adjacent BLACK nodes in the E and S directions. These BLACK–WHITE and WHITE–BLACK pairs make a contribution to the perimeter of the image. It should be clear that the two methods are equivalent. However, depending on the particular configuration of BLACK and WHITE blocks in the image, one method may be superior to the other. In particular, if there are considerably more BLACK nodes than WHITE nodes, then the second method may be preferable since less adjacencies need to be explored.

The algorithm demonstrates the utility of the quadtree as a desirable data structure for image representation. Computation of perimeter is generally achieved by use of a chain code representation. We have shown that it can be computed with reasonable efficiency when the quadtree is used as the data structure.

Acknowledgment

The author would like to thank K. Riley for her help in preparing this manuscript and P. Young for drawing the figures. He has also benefited greatly from discussions with C. R. Dyer and A. Rosenfeld.

References

[1] C. R. Dyer, A. Rosenfeld, and H. Samet, "Region representation: Boundary codes from quadtrees," *Commun. Ass. Comput. Mach.*, vol. 23, pp. 171–179, Mar. 1980.

[2] H. Freeman, "Computer processing of line-drawing images," *Comput. Surveys*, vol. 6, pp. 57–97, 1974.

[3] G. M. Hunter, "Efficient computation and data structures for graphics," Ph.D. dissertation, Dep. Elec. Eng. Comput. Sci., Princeton Univ., Princeton, NJ, 1978.

[4] G. M. Hunter and K. Steiglitz, "Operations on using quadtrees," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, pp. 145–153, 1979.

[5] ——, "Linear transformation of pictures represented by quadtrees," *Comput. Graphics Image Processing*, vol. 10, pp. 289–296, 1979.

[6] A. Klinger, "Patterns and search statistics," in *Optimizing Methods in Statistics*, J. S. Rustagi, Ed. New York: Academic, 1971.

[7] A. Klinger and C. R. Dyer, "Experiments in picture representation using regular decomposition," *Comput. Graphics Image Processing*, vol. 5, pp. 68–105, 1976.

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

[9] P. Naur, Ed., "Revised report on the algorithmic language Algol 60," *Commun. Ass. Comput. Mach.*, vol. 3, pp. 299–314, 1960.

[10] A. Rosenfeld and A. C. Kak, *Digital Picture Processing.* New York: Academic, 1976, section 9.2.1.

[11] H. Samet, "Region representation: Quadtrees from boundary codes," *Commun. Ass. Comput. Mach.*, vol. 23, pp. 163–170, Mar. 1980.

[12] ——, "Connected component labeling using quadtrees," *J. Ass. Comput. Mach.*, vol. 28, pp. 487–501, July 1981.