A Model for the Analysis of Neighbor Finding in Pointer-Based Quadtrees

HANAN SAMET AND CLIFFORD A. SHAFFER

Abstract—**A natural byproduct of the tree-like nature of the quadtree is that many basic image processing operations can be implemented as tree traversals which differ in the nature of the computation that is performed at each node. Some of these computations involve the inspection of a node's adjacent neighbors (termed neighbor finding). A new model is developed for images represented by quadtrees, and it is used to analyze various neighbor-finding techniques. The model's predicted costs for neighbor finding correlate very closely with empirical results and it is superior to the model that was used previously.**

Index Terms—**Cartography, computer graphics, hierarchical data structures, image processing, neighbor finding, quadtrees.**

I. Introduction

In recent years, the region quadtree representation [6], [12] has gained use as a data structure for applications in image processing, computer graphics, cartography, and other areas. A natural byproduct of the tree-like nature of the quadtree is that many basic operations can be implemented as tree traversals (e.g., connected component labeling [9], etc.). The difference between them lies in the nature of the computation that is performed at the node. Often, these computations involve the examination of nodes whose corresponding blocks are adjacent to the block whose node is being processed. We call such nodes *neighbors*, and the process of locating them is termed *neighbor finding*. Neighbor finding is also crucial to algorithms for converting between representations (e.g., to construct a quadtree from a boundary code [8] and vice versa [3], etc.).

Manuscript received September 14, 1984; revised May 28, 1985. Recommended for acceptance by S. L. Tamimoto. This work was supported by the National Science Foundation under Grant MCS-83-02118.

The authors are with the Department of Computer Science and the Center for Automation Research, University of Maryland, College Park, MD 20742.

In [10], algorithms are described and analyzed for locating different kinds of neighbors. The analysis of the execution time of the algorithms was done in terms of the average number of nodes (using a particular image generation model) that needed to be traversed in order to locate the desired neighbor. The contributions of this correspondence are an improved model and empirical results that are in very close correlation with the model. For our new model, the observed values are within 6 percent of the predicted values, whereas for the old model, the predicted values were larger than the observed values by between 21 and 37 percent.

II. Definitions

The quadtree is an approach to region representation that is based on the successive subdivision of an image array into quadrants. If the array does not consist entirely of 1's or entirely of 0's (i.e., the region does not cover the entire array), then we subdivide it into quadrants, subquadrants, until we obtain square blocks (possibly single pixels) that consist entirely of 1's or entirely of 0's; i.e., each block is entirely contained in the region or entirely disjoint from it. As an example, consider the region shown in Fig. 1(a) which is represented by the $2^3 \times 2^3$ binary array in Fig. 1(b). The resulting blocks for the array of Fig. 1(b) are shown in Fig. 1(c) and the corresponding quadtree is shown in Fig. 1(d). All examples cited in this correspondence refer to Fig. 1. For a $2^n \times 2^n$ image, we say that the root is at level n, and that a node at level i is at a distance of $n - i$ from the root of the tree. Nodes corresponding to pixels are at level 0.

Each node of a quadtree corresponds to a block in the original image. We use the terms block and node interchangeably. The term that will be used depends on whether we are referring to a block decomposition [i.e., Fig. 1(c)] or a tree [i.e., Fig. 1(d)]. Each block has four sides and four corners. At times we speak of sides and corners collectively as directions. Let the four sides of a node's block be called its N, E, S, and W sides. The four corners of a node's block are labeled NW, NE, SW, and SE. Given two nodes P and Q whose corresponding blocks do not overlap, and a direction D, we define a predicate *adjacent* such that *adjacent* (P, Q, D) is true if there exist two pixels p and q contained in P and Q, respectively, such that either q is adjacent to side D of p, or corner D of p is adjacent to the opposite corner of q. In such a case, we say that nodes P and Q are *neighbors*. For example, nodes J and 39 in Fig. 1 are neighbors since J is to the west of 39; similarly, for nodes 38 and H since H is to the northeast of 38. In this correspondence, we are interested in the following four types of neighbors. In the construction of names we use the following correspondence: G for "greater than or equal," C for "corner," S for "side," and N for "neighbor."

1) $GSN(P, D) = Q$: Node Q corresponds to the smallest block (it may be gray) adjacent to side D of node P of size greater than or equal to the block corresponding to P.

2) $CSN(P, D, C) = Q$: Node Q corresponds to the smallest block that is adjacent to the D side of the C corner of node P.

3) $GCN(P, C) = Q$: Node Q corresponds to the smallest block (it may be gray) opposite to C corner of node P of size greater than or equal to the block corresponding to P.

4) $CCN(P, C) = Q$: Node Q corresponds to the smallest block that is opposite to the C corner of node P.

For example, in Fig. 1, $GSN(J, E) = K$, $GSN(J, S) = L$, $CSN(J, E, SE) = 39$, $GCN(H, NE) = G$, $GCN(H, SW) = K$, and $CCN(H, SW) = 38$.

III. Neighbor Finding

The relations *GSN*, *CSN*, *GCN*, and *CCN* are implemented in [10] using a quadtree implementation where each node has four links from a node to its four sons and one link to its father for a nonroot node. For example, to compute $GSN(P, D)$, the basic idea is to ascend the tree until a common ancestor of both P and $GSN(P, D)$ is located, and then descend the tree in search of $GSN(P, D)$.

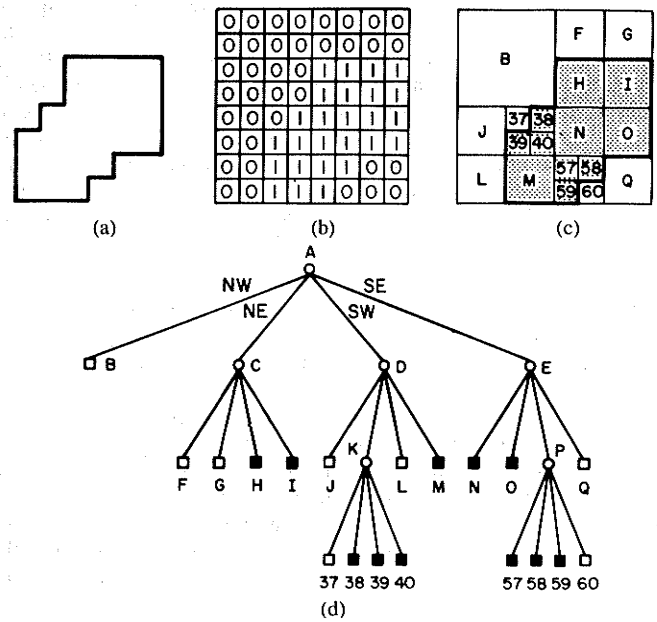

Fig. 1. A region (a), its binary array (b), and its maximal blocks [block decomposition of the region in (a)]—blocks in the region are shaded. (d) Corresponding quadtree representation of the blocks in (c).

Clearly, we can always ascend as far as the root of the tree and then start our descent. However, our goal is to find the nearest common ancestor as this minimizes the number of nodes that must be visited. In order to see how this process works, suppose that we wish to determine *GSN*(M, E) in Fig. 1. The nearest common ancestor is the first ancestor node which is reached via its *NW* or *SW* son (i.e., the first ancestor node of which M is not an eastern descendant). Next, we retrace the path used to locate the nearest common ancestor, except that we make mirror image moves about an axis formed by the common boundary between the nodes. In the case of an eastern neighbor, the mirror images of *NE* and *SE* are *NW* and *SW*, respectively, Therefore *GSN*(M, E) = P. It is located by ascending the tree until the nearest common ancestor A has been reached. This requires going through an *SE* link to reach node D, and an *SW* link to reach node A. Node P is subsequently located by backtracking along the previous path with the appropriate mirror image moves (i.e., following an *SE* link to reach node E, and an *SW* link to reach node P). Similar techniques are described in [10] to compute *CSN*, *GCN*, and *CCN*.

IV. Analysis

The average execution time of the functions *GSN*, *CSN*, *GCN*, and *CCN* is analyzed in [10] in terms of the number of nodes that must be visited in locating the desired neighbor. The analysis of each function can be decomposed into two stages corresponding to the process of locating the nearest common ancestor, and then locating the desired neighbor. A random image model was used under which each node is assumed to be equally likely to appear at any position and level in the quadtree. An equivalent statement is that $\frac{3}{4} \cdot (\frac{1}{4})^i$ of the nodes are at level i. Observe that our notion of a random image differs from the conventional one which implies that every pixel has an equal probability of being black or white. Such an assumption leads to a very low probability of aggregation (i.e., nodes corresponding to blocks of size greater than 1 pixel). Clearly, for such an image the quadtree is the wrong representation (e.g., a checkerboard). The problem with the conventional random image model is that it assumes independence which is clearly not the case (i.e., a pixel's value is typically related to that of its neighbors). To see this more vividly, consider Table I which tabulates the distribution of blocks of varying size for five images that are described in Section V. Nevertheless, there is room for a better theoretical justification for our model.

In order to analyze the second stage, we must also model the distribution of neighbor pairs (i.e., the possible configurations of adjacent nodes of varying nodes of varying sizes). There is a number of models to choose from. In this correspondence, we present a new model and show how it correlates very closely with empirical results (see Section V). For example, suppose that we wish to determine *GSN*(59, W) (i.e., M). In theory, there are three possible neighbors—i.e., one each of size 1×1, 2×2, and 4×4 at node distances of 6, 5, and 4, respectively. To compute the average value of *GSN*, the model employed in [10] termed the *old model*, treats each of these cases individually and as equally probable—i.e., a node in the same position as 59 makes three contributions to the average value. In contrast, the model which we introduce and use in this correspondence, termed the *new model*, only includes the average contribution of these three cases. This change is justified by the fact that, although the neighbor of a large node may appear at any of several possible levels, only one neighbor will actually exist. While this modification of the model seems trivial, tests on complex images show that its use leads to a very close correlation between theory and practice.

TABLE I
Node Size (Percent) Distribution

Leaf Node Size	Model %	Flood	Topo	Land	Stone	Pebble
1 by 1	75.00	2,468 (47.4)	14,832 (59.3)	16,112 (56.4)	19,732 (61.7)	27,316 (60.8)
2 by 2	18.75	1,559 (29.9)	7,336 (29.3)	8,484 (29.7)	8,219 (25.7)	11,995 (26.7)
4 by 4	4.69	660 (12.7)	2,175 (8.70)	2,984 (10.5)	2,784 (8.71)	4,418 (9.83)
8 by 8	1.17	263 (5.05)	470 (1.88)	784 (2.62)	974 (3.05)	1,095 (3.44)
16 by 16	.293	175 (3.36)	138 (.552)	176 (.613)	213 (.666)	108 (.240)
32 by 32	.073	57 (1.09)	51 (.204)	38 (.133)	47 (.147)	18 (.040)
64 by 64	.018	22 (.423)	8 (.032)	8 (.028)	0 (0.00)	0 (0.00)
128 by 128	.046	2 (.038)	2 (.008)	0 (0.00)	0 (0.00)	0 (0.00)
Total		5,206	25,012	28,549	31,969	44,950

We first present an analysis of stage one (i.e., the number of nodes that must be visited in locating the nearest common ancestor for side and corner directions) as it is common to *GSN*, *GCN*, *CSN*, and *CCN*. This analysis is identical to that performed in [10]. Next, we analyze stage two. It is here that we employ the new model for the distribution of neighbor pairs.

Theorem 1: The average number of nodes that must be visited in locating the nearest common ancestor when seeking a neighbor in a side direction is bounded from above by 2.

Proof: Given a node P at level i and a direction toward side D, there are $2^{n-i} \cdot (2^{n-i} - 1)$ possible positions where P might be located. Of these positions, $2^{n-i} \cdot 2^0$ 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 $i + 1$. The same analysis is repeated for each level and all the possible cases are treated as equally likely. Starting at a node at level i, to reach a nearest common ancestor at level j, $j - i$ nodes must be visited. Therefore, the average as $n(n \geq 1)$ gets large is

$$\frac{\sum_{i=0}^{n-1} \sum_{j=i+1}^{n} 2^{n-i} \cdot 2^{n-j} \cdot (j-i)}{\sum_{i=0}^{n-1} 2^{n-i} \cdot (2^{n-i} - 1)} = 2 - \frac{6 \cdot (n-1) \cdot 2^n + 6}{2^{2n+2} - 6 \cdot 2^n + 2} \leq 2.$$

Theorem 2: The average number of nodes that must be visited in locating the nearest common ancestor when seeking a neighbor in a corner direction is bounded from above by $\frac{8}{3}$.

Proof: Given a node P at level i and a direction toward quadrant C, there are $(2^{n-i} - 1)^2$ possible positions where P might be located. In [10] it is shown that of these positions $4^0 \cdot (2 \cdot (2^{n-i} - 1) - 1)$ have neighbors in direction C such that the nearest common ancestor is at level n, $4^1 \cdot (2 \cdot (2^{n-i-1} - 1) - 1)$ at level $n - 1, \cdots$, and $4^{n-i-1} \cdot (2 \cdot (2^{n-i-(n-i-1)} - 1) - 1)$ at level $i + 1$. The same analysis is repeated for each level and all the possible cases are treated as equally likely. Starting at a node at level i, to reach a nearest common ancestor at level j, $j = i$ nodes must be visited. Therefore, the average as $n(n \geq 1)$ gets large is

$$\frac{\sum_{i=0}^{n-1}\sum_{j=i+1}^{n} 4^{n-j}\cdot(2\cdot(2^{n-i-(n-j)}-1)-1)\cdot(j-i)}{\sum_{i=0}^{n-1}(2^{n-1}-1)^2} = \frac{8}{3}-\frac{(6n-10)\cdot 2^{n+2}-3n^2+5n+40}{2^{2n+3}-3\cdot 2^{n+3}+6n+16} \le \frac{8}{3}.$$

•

The results of theorems 1 and 2 can be used to compute the average number of nodes visited in locating neighbors in a checkerboard image or any image represented by a full quadtree since each neighbor is found at the same level as the starting node. In particular, the average number of nodes visited in locating a vertical or horizontal neighbor is 4 while it is $\frac{16}{3}$ for a neighbor in the direction of a corner. When the analysis performed above for theorem 1 is restricted to one row, then we have analyzed the average cost of accessing adjacent elements in a two-dimensional array that has been embedded in a binary tree [2] (although there it is analyzed by techniques that make use of recurrence relations). To determine the average cost of finding neighbors of greater than or equal size in the horizontal and vertical directions (i.e., *GSN*) we use theorem 3 given below. Theorem 3 reflects the cost of the first and second stages. The cost of the second stage is obtained by subtracting the result of theorem 1 from that of theorem 3.

Theorem 3: The average number of nodes visited by *GSN* is bounded from above by $\frac{7}{2}$.

Proof: Using similar reasoning as in theorem 1, given a node P at level i, a direction D, and having a nearest common ancestor at level j, there are $j-i$ possible neighbors of size greater than or equal to that of P. Therefore, the average number of nodes that will be visited in the process of locating such a neighbor is $j-i+1/(j-i)\cdot\Sigma_{k=i}^{j-1}(j-k)$. This is obtained by observing that the nearest common ancestor is at a distance of $j-i$, while each of the possible neighbors is at level k where k ranges between i and $j-1$, and at a distance of $j-k$ from the nearest common ancestor. Therefore, the average number of nodes visited by *GSN* as $n(n \ge 1)$ gets large is

$$\frac{\sum_{i=0}^{n-1}\sum_{j=i+1}^{n} 2^{n-i}\cdot 2^{n-j}\cdot\left(j-i+\frac{1}{j-i}\cdot\sum_{k=i}^{j-1}(j-k)\right)}{\sum_{i=0}^{n-1}2^{n-i}\cdot(2^{n-i}-1)} = \frac{7}{2}-\frac{9\cdot(n-1)\cdot 2^n+9}{2^{2n+2}-6\cdot 2^n+2} \le \frac{7}{2}.$$

•

Theorems 4, 5, and 6 yield the average number of nodes visited by *GCN*, *CSN*, and *CCN*, respectively, and are derived in a manner analogous to theorem 3.

Theorem 4: The average number of nodes visited by *GCN* is bounded from above by $\frac{9}{2}$.

Theorem 5: The average number of nodes visited by *CSN* is bounded from above by $\frac{11}{3}$.

Theorem 6. The average number of nodes visited by *CCN* is bounded from above by $\frac{14}{3}$.

In order to complete the comparison to [10], we also recompute the average number of nodes visited by *GSN* in a roped quadtree [4] and by procedure ALIGNED. Recall that in a roped quadtree, there exist explicit links between adjacent nodes in the horizontal and vertical directions. In particular, a rope is a link between two adjacent nodes of equal size where at least one of the nodes is a leaf node. Procedure ALIGNED (see [10]) is used to determine if edges of adjacent nodes extend past each other or are aligned. Theorems 7 and 8 are derived in a manner analogous to the second stage of theorem 3. The difference between the results of theorems 7 and 8 is that for ALIGNED there is no need to traverse a link corresponding to a rope. For more detailed proofs of all the theorems, see [13].

Theorem 7: The average number of nodes visited by *GSN* in a roped quadtree is bounded from above by $\frac{3}{2}$.

Theorem 8: The average number of nodes visited by ALIGNED is bounded from above by $\frac{1}{2}$.

V. Empirical Results

In order to test the validity of the model, we conducted a number of experiments with images corresponding to a cartographic database as well as some standard textures. In particular, we used five 512×512 images which were readily available. Three of them correspond to a land-use map, and a floodplain map of a region in Northern California [11]. We also selected two binary images having sufficient complexity to make them of interest. They correspond to a pebble texture (*D*23) and a stone texture (*D*7) [1]. All of the images were represented using quadtrees and each of the neighbor finding functions described above was applied in all four directions at each leaf node in the images. In the tables that follow, the maps are arranged in ascending order of complexity where complexity is the number of nodes in the image. All references to model values are for a depth of $n = 9$. Table II summarizes the observed values for the individual images, the average value over all five images, the value predicted by the new model, and the value predicted by the old model.

From Table II we see that values predicted by the old model were between 21 and 37 percent above the observed values. In contrast, values predicted by the new model are within 6 percent of the observed values. We can get a more accurate evaluation of the new model by recalling that the neighbor finding process can be decomposed into two stages. The first stage locates the nearest common ancestor. The old and new models do not differ in the analysis of this stage. There are two cases depending on whether we are seeking a neighbor in the side or corner direction. Table III shows the empirical results. It is interesting to note how close the model correlates with the observed values.

Table IV gives the cost of the second stage of the neighbor finding process. This stage reflects use of our new model and a comparison with the old model reveals the improvement. In particular, we see that for the second stage the values predicted by the old model were between 50 and 89 percent above the observed values whereas the observed values are within 10 percent of that predicted by the new model. This is much more reasonable and reinforces use of the new model. Perhaps the most important feature of the new model is the intuitively correct prediction that locating neighbors that are of greater than or equal size is cheaper than finding neighbors of equal size. The cost of the latter is simply twice the cost of locating a nearest common ancestor. The reason for the poor performance of the old model was that all of a node's possible neighbors were individually taken into account when computing the average, whereas the new model only included the average contribution of the possible neighbors (see Section IV).

In order to improve our understanding of the difference between the predicted and observed values of the first and second stages of the neighbor finding process, we tabulate in Tables V and VI the average cost of *CSN* in all four directions as a function of the distance to and from, respectively, the nearest common ancestor. We use *CSN* because it provides the greatest range of values for testing the model used in the second stage. Remember that our model assumes that $\frac{3}{4}\cdot(\frac{1}{4})^i$ of the nodes are at level i. For each node at level i, its nearest common ancestor is at level $j(n \ge j > i)$ with probability $(\frac{1}{2})^{j-i}$. This is relevant to the analysis of the first stage. The average number of nodes visited in locating the neighbor when starting at the nearest common ancestor is $(j+1)/2$. This is relevant to the analysis of the second stage. The final row of the table accounts for the nodes that are adjacent to the border of the image and so have no neighbor in the tree.

Table V shows the observed distribution of the nodes that are at a distance i from their nearest common ancestor for the first stage. The observed values are in close agreement with the predicted val-

TABLE II
AVERAGE COST OF NEIGHBOR FINDING OPERATIONS

Operation	Observed						New Model	Old Model
	Flood	Topo	Land	Stone	Pebble	Average		
GSN	3.50	3.60	3.59	3.58	3.56	3.57	3.46	4.79
CSN	3.66	3.75	3.73	3.73	3.71	3.72	3.63	4.89[a]
GCN	4.47	4.68	4.63	4.64	4.60	4.60	4.44	5.77
CCN	4.64	4.83	4.79	4.79	4.75	4.76	4.60	5.85

[a]In [10], the limit value is erroneously computed to be $\frac{14}{3}$.

TABLE III
AVERAGE COST OF LOCATING THE NEAREST COMMON ANCESTOR

Type of Neighbor	Observed						Model
	Flood	Topo	Land	Stone	Pebble	Average	
side	2.01	2.00	2.00	2.00	1.99	2.00	1.98
corner	2.69	2.67	2.66	2.66	2.65	2.67	2.62

TABLE IV
AVERAGE COST OF LOCATING THE NEIGHBOR STARTING AT THE NEAREST COMMON ANCESTOR

Operation	Observed						New Model	Old Model
	Flood	Topo	Land	Stone	Pebble	Average		
GSN	1.49	1.60	1.59	1.58	1.57	1.57	1.48	2.81
CSN	1.64	1.74	1.73	1.73	1.72	1.71	1.65	2.91
GCN	1.79	2.00	1.97	1.98	1.95	1.94	1.82	3.15
CCN	1.98	2.15	2.13	2.13	2.10	2.09	1.98	3.23

TABLE V
NODE SIZE (PERCENT) DISTRIBUTION FOR STAGE ONE OF THE NEIGHBOR FINDING PROCESS

Distance from node to nca	Model	Observed					
		Flood	Topo	Land	Stone	Pebble	Average
1	50.00	50.00	50.00	50.00	50.00	50.00	50.00
2	25.00	25.48	25.17	25.02	25.00	24.95	25.12
3	12.50	11.81	12.23	12.42	12.54	12.52	12.26
4	6.25	6.08	6.10	6.37	6.19	6.43	6.23
5	3.13	2.86	3.00	2.99	3.07	3.11	3.01
6	1.56	2.41	1.72	1.62	1.58	1.51	1.77
7	0.78	0.69	0.94	0.83	0.90	0.82	0.84
8	0.39	0.54	0.56	0.51	0.48	0.40	0.50
9	0.20	0.23	0.22	0.20	0.21	0.18	0.21
On Edge	0.20	0.10	0.06	0.03	0.04	0.06	0.06
Expected Cost	2.00	2.01	2.00	2.00	2.00	1.99	2.00

TABLE VI
AVERAGE COST OF STAGE TWO OF THE NEIGHBOR FINDING PROCESS

Depth of nca	Model		Observed											
			Flood		Topo		Land		Stone		Pebble		Average	
	Cost	%	Cost	%	Cost	%	Cost	%	Cost	%	Cost	%	Cost	%
1	1	37.50	1.00	23.66	1.00	29.65	1.00	28.22	1.00	30.86	1.00	30.36	1.00	28.66
2	1.5	28.13	1.28	26.76	1.31	29.49	1.32	28.97	1.36	28.29	1.34	28.64	1.32	28.41
3	2	16.41	1.65	19.02	1.77	19.07	1.73	19.70	1.81	18.61	1.74	19.17	1.73	19.21
4	2.5	8.79	2.02	12.28	2.49	10.41	2.34	11.25	2.35	10.76	2.30	10.89	2.31	11.12
5	3	4.54	2.31	7.04	3.31	5.33	3.14	5.85	3.10	5.68	3.29	5.62	3.03	6.02
6	3.5	2.30	2.94	4.39	4.17	2.77	4.03	2.88	3.84	2.91	4.00	2.67	3.81	3.12
7	4	1.16	2.90	2.83	5.08	1.84	5.01	1.66	5.00	1.49	5.21	1.50	4.97	1.76
8	4.5	0.58	4.49	1.64	6.66	0.95	5.90	0.90	6.22	0.85	6.36	0.74	6.72	1.00
9	5	0.29	0.22	0.76	7.18	0.48	7.32	0.45	7.13	0.45	7.17	0.37	7.00	0.50
On Edge		0.50		0.60		0.22		0.23		0.21		0.22		0.28
Expected Cost	1.66		1.84		1.74		1.73		1.73		1.72		1.72	

ues. In fact, for a distance of 1 the predicted and observed percentages are exactly the same. This is not surprising because we tabulated *CSN* in all four directions and each node has two brothers at a distance of 1. For all of the images the average distance to the nearest common ancestor is within two percent of the predicted value.

Table VI shows the average number of nodes that must be descended from a nearest common ancestor at level i in the second stage before encountering the desired neighbor for *CSN* as well as the distribution of the nodes having a nearest common ancestor at level i. Again, the observed values are in close agreement with the predicted values. The observed values are in general slightly higher than the predicted values because there is a greater percentage of nodes whose distance from the nearest common ancestor exceeds the predicted distance. Thus, their contribution is weighted more heavily thereby increasing the average cost. Notice that the floodplain image has an observed cost which is lower than the predicted cost. This is not surprising because it has a greater preponderance of large blocks than the other images (see Table I) as well as that predicted by the model. The latter can be seen by noting that the average distances shown in Table VI are almost always less than those predicted by our model. The remaining images have more smaller blocks and the average distance to the neighbor from the nearest common ancestor often exceeds the average used in the model.

VI. CONCLUDING REMARKS

A new analysis has been presented of the costs of neighbor finding methods in pointer-based quadtree representations. It correlates strongly with empirical observations. It reinforces our prior conclusion that use of neighbor finding is preferable to using links between neighbors [4] which has a higher storage requirement although its expected execution time is lower. In addition, we have shown that quadtree images can be modeled in a way that permits the analysis of algorithms that operate on them. Although we have attempted to explain why our model works, a more "theoretical" explanation is still lacking.

Our new results impact previously published analyses of algorithms involving conversion between representations (e.g., quadtrees and boundary codes [3], [8], etc.) as well as basic image operations (e.g., connected component labeling [9]). In the case of the former, neighbor finding is crucial to the performance of the task while in the latter there also exist implementations that do not require neighbor finding (e.g., transmitting the neighbors as an argument [5], [14]). Nevertheless, the orders of the expected execution times of the algorithms remain the same.

ACKNOWLEDGMENT

We have benefited greatly from the comments of R. E. Webber.

REFERENCES

[1] P. Brodatz, *Textures*. New York: Dover, 1966.
[2] R. A. DeMillo, S. C. Eisenstat, and R. J. Lipton, "Preservng average proximity in arrays," *Commun. Ass. Comput. Mach.*, vol. 21, no. 3, pp. 228–231, Mar. 1978.
[3] C. R. Dyer, A. Rosenfeld, and H. Samet, "Region representation: Boundary codes from quadtrees," *Commun. Ass. Comput. Mach.*, vol. 23, no. 3, pp. 171–179, Mar. 1980.
[4] G. M. Hunter and K. Steiglitz, "Operations on images using quadtrees," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-1, pp. 145–153, Apr. 1979.
[5] C. Jackins and S. L. Tanimoto, "Quad-trees, oct-trees, and k-trees—A generalized approach to recursive decomposition of Euclidean space," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-5, pp. 533–539, Sept. 1983.
[6] A. Klinger, "Patterns and search statistics," in *Optimizing Methods in Statistics*, J. S. Rustagi, Ed. New York: Academic, 1971, pp. 303–337.
[7] A. Klinger and M. L. Rhodes, "Organization and access of image data by areas," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-1, pp. 50–60, June 1979.
[8] H. Samet, "Region representation: Quadtrees from boundary codes," *Commun. Ass. Comput. Mach.*, vol. 23, no. 3, pp. 163–170, Mar. 1980.
[9] ——, "Connected component labeling using quadtrees," *J. Ass. Comput. Mach.*, vol. 28, no. 3, pp. 487–501, July 1981.
[10] ——, "Neighbor finding techniques for images represented by quadtrees," *Comput. Graph. Image Processing*, vol. 18, no. 1, pp. 37–57, Jan. 1982.
[11] H. Samet, A. Rosenfeld, C. Shaffer, and R. E. Webber, "Quadtree region representation in cartography: Experimental results," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, pp. 1148–1154, Nov. 1983.
[12] H. Samet, "The quadtree and related hierarchical data structures," *ACM Comput. Surveys*, vol. 16, no. 2, pp. 187–260, June 1984.
[13] H. Samet and C. A. Shaffer, "A model for the analysis of neighbor finding in pointer-based quadtrees," Dep. Comput. Sci., Univ. Maryland, College Park, Tech. Rep. TR-1432, Aug. 1984.
[14] H. Samet, "A top-down quadtree traversal algorithm," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-7, pp. 94–98, Jan. 1985.