Image Processing and Computer Vision

Robert M. Haralick Editor

Data Structures for Quadtree Approximation and Compression

HANAN SAMET

ABSTRACT: A number of data structures for representing images by quadtrees without pointers are discussed. The image is treated as a collection of leaf nodes. Each leaf node is represented by use of a locational code corresponding to a sequence of directional codes that locate the leaf along a path from the root of the tree. Somewhat related is the concept of a forest which is a representation that consists of a collection of maximal blocks. It is reviewed and refined to enable the representation of a quadtree as a sequence of approximations. In essence, all BLACK and WHITE nodes are said to be of type GB and GW, respectively. GRAY nodes are of type GB if at least two of their sons are of type GB; otherwise, they are of type GW. Sequences of approximations using various combinations of locational codes of GB and GW nodes are proposed and shown to be superior to approximation methods based on truncation of nodes below a certain level in the tree. These approximations have two important properties. First, they are progressive in the sense that as more of the image is transmitted, the receiving device can construct a better approximation (contrast with facsimile methods which transmit the image one line at a time). Second, they are proved to lead to compression in the sense that they never require more than MIN(B, W) nodes where B and W correspond to the number of BLACK and WHITE nodes in the original quadtree. Algorithms are given for constructing the approximation sequences as well as decoding them to rebuild the original quadtree.

This work was supported in part by NSF Grant DCR-83-02118.

1. INTRODUCTION

Region representation is an important issue in graphics and image processing applications. A number of representations are currently in use. These have been traditionally dominated by run lengths, binary arrays, and chain codes. Recently, the quadtree of Klinger [12, 24] has received considerable attention as a data structure for such applications. This renewed interest was due in a large measure to the work of Hunter and Steiglitz in the domain of computer graphics and animation [6–8], and work showing the interchangeability of the quadtree representation with other representations such as boundary codes [4, 19] and rasters [20]. A variant of the quadtree, termed an *octree*, has also been used for three-dimensional data [6, 9, 15, 30].

The quadtree was originally conceived as an alternative to binary array image representation with the goal of saving space by aggregation of similar regions. Of course, if there is no aggregation (e.g., a checkerboard image), then the quadtree will require more space than the binary array. However, more importantly, the hierarchical nature of the quadtree also results in savings in execution time. In particular, algorithms using quadtrees have execution times that are only dependent on the number of blocks in the image and not on their size (e.g., connected component labeling [21]). Nevertheless, by virtue of the tree structure, the amount of space required for pointers from a node to its sons is not trivial. As a result, there has recently been a considerable amount of interest in pointerless quadtree repre-

sentations. They can be grouped into two categories. The first represents the image in the form of a preorder traversal of the nodes of its quadtree [11]. The second treats the image as a collection of leaf nodes. Each leaf is represented by use of a locational code corresponding to a sequence of directional codes that locate the leaf along a path from the root of the tree. It has a large number of variants and is used by many researchers [1–3, 5, 13, 16, 17, 28, 29]. Somewhat closely related is the concept of a forest of quadtrees of Jones and Iyengar [10] which uses a variant of a locational code to represent the roots of the elements of the forest.

In this article, we define a sequence of approximations to quadtrees that are based on the notions of forests. In essence, all BLACK and WHITE nodes are said to be of type GB and GW, respectively. GRAY nodes are of type GB if at least two of their sons are of type GB; otherwise, they are of type GW. Thus, we see that this redefinition of a GRAY node does not cost any extra storage. We propose a number of approximation sequences using various combinations of the locational codes of GB and GW nodes and show them to be superior to the inner and outer approximations discussed in [18]. Our sequences of approximations possess two important properties. First, they are progressive. This means that as more of the image is transmitted, the receiving device can construct a better approximation. Progressive approximation methods are to be contrasted with facsimile methods that transmit the image one line at a time. Second, use of our approximation methods also leads to image compression in the sense that the image never requires more nodes when our approximation methods are used. In fact, they never lead to fewer nodes than are required were the image to be represented solely by BLACK nodes or solely by WHITE nodes.

This article is organized as follows. In the remainder of this section we give a brief definition of a quadtree. Section 2 contains a discussion of the pointerless quadtree representation that we use. In particular, we elaborate on a number of different ways of defining a locational code and motivate our choice. Section 3 is a brief overview of hierarchical approximation methods. Sections 4 and 5 constitute the heart of the article. In Section 4 we define the concept of a forest-based approximation method and in Section 5 we discuss the degree of image compression that is achievable when our approximation methods are used. Section 6 presents experimental results from the use of our approximation methods and an interpretation of their significance. Section 7 concludes our presentation with a set of open questions for future research.

Given a $2^n \times 2^n$ array of unit pixels, a quadtree is constructed by repeatedly subdividing the array into quadrants, subquadrants, . . . until we obtain blocks which consist of a single value (e.g., a gray level). This process is represented by a tree of out degree 4 in which the root node corresponds to the entire array, the four sons of the root node correspond to the quadrants, and the terminal nodes correspond to those blocks of the array for which no further subdivision is necessary. The nodes at level k (if any) represent blocks of size $2^k \times 2^k$ and are often referred to as nodes of size 2^k. Thus, a node at level 0 corresponds to a single pixel in the image, while a node at level n is the root of the quadtree. For example, Figure 1b is a block decomposition of the region in Figure 1a while Figure 1c is the corresponding quadtree. In general, we will be dealing with two values 1 and 0 where BLACK and WHITE square nodes in the tree represent blocks consisting entirely of 1's and 0's, respectively. Circular nodes, also termed GRAY nodes, denote nonterminal nodes.

Each node in a quadtree is stored as a record containing six fields. The first five fields contain pointers to the node's father and its four sons, labeled NW, NE, SW, and SE. Given a node P and a son I, these fields are referenced as FATHER(P) and SON(P, I), respectively. We can determine the specific quadrant in which a node, say P, lies relative to its father by use of the function SONTYPE such that SONTYPE(P) = I if SON(FATHER(P), I) = P. The sixth field, NODETYPE, describes the contents of the block of the image which the node represents, that is, BLACK, WHITE, or GRAY.

2. POINTERLESS QUADTREE REPRESENTATIONS

Instead of using a tree consisting of BLACK, WHITE, and GRAY nodes, we use a representation in the form of a collection of the leaf nodes of the quadtree. Each leaf node is represented by a locational code which corresponds to an encoding of the path from the root of the tree to the node. In fact, we only need to maintain a collection of the BLACK nodes since we represent binary images and the only GRAY nodes are internal to the data structure which means that the WHITE nodes can be determined given the BLACK nodes. As mentioned in Section 1, there exist many variants of the locational code. In this article, we make use of the following definition of a locational code. Let the sequence $\langle x_i \rangle$ represent the path of nodes from the root of a quadtree to x_m, the desired node, such that x_n = root of the quadtree and x_i = FATHER(x_{i-1}). The directions NW, NE, SW, and SE are represented by the directional codes 1, 2, 3, and 4, respectively, and are accessed by the function SONTYPE5. The encoding of the locational code for node x_m is given by z_n where z_i is defined:

$$z_i = \begin{cases} 0 & i = m \\ 5 \cdot z_{i-1} + \text{SONTYPE5}(x_i) & m < i \leq n. \end{cases}$$

For example, node 10 of Figure 1 would be encoded by the number $z_4 = 582$. It can be decoded into the sequence of directional codes $\langle b_i \rangle = \langle 2, 1, 3, 4 \rangle$ = ⟨NE, NW, SW, SE⟩—that is, $z_4 = 4 \cdot 5^3 + 3 \cdot 5^2 + 1 \cdot 5^1 + 2 \cdot 5^0$.

The above encoding method has a number of useful features. First, it lends itself easily to decoding a locational code into the sequence of directional codes by

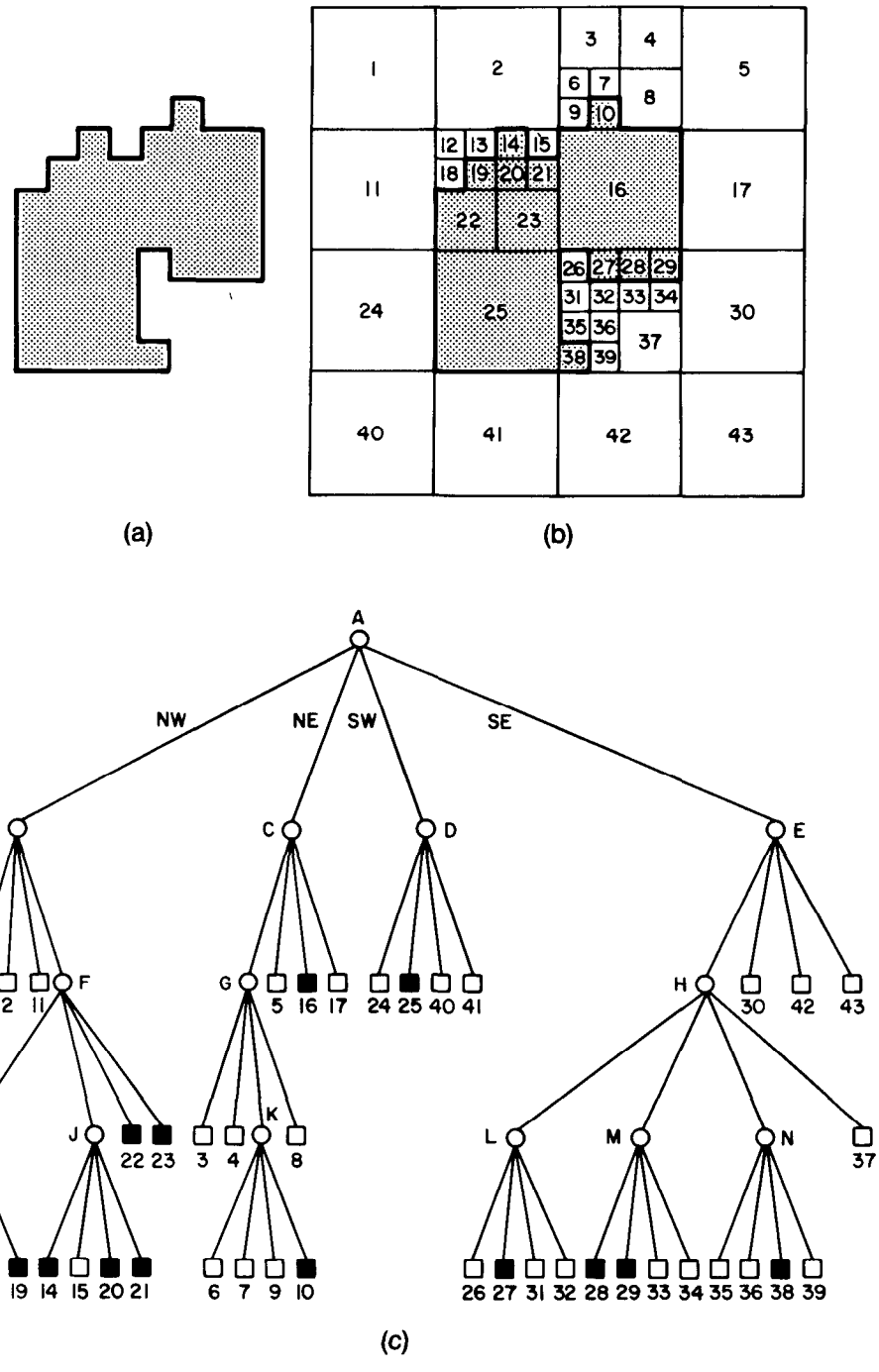


FIGURE 1. A Region, its Maximal Blocks, and the Corresponding Quadtree. Blocks in the region are shaded, background blocks are black. (a) Region. (b) Block decomposition of the region in (a). (c) Quadtree representation of the blocks in (b).

using a combination of modulo and integer division operations. This is an important operation and must be fast and computationally simple. In essence, we are able to decode the number in such a way that we obtain the directional codes in the order in which we traverse a path from the root of the quadtree to the root of the subquadtree. Second, sorting the codes of the BLACK nodes in increasing order, results in a list which is a variant of a breadth-first traversal of the BLACK portion of the tree, that is, for $i < j$, BLACK nodes at level j will appear in the list before BLACK nodes at level i. For example, listing the BLACK nodes of Figure 1, in increasing order of locational codes, yields 25, 16, 22, 23, 28, 14, 27, 29, 20, 38, 19, 21, and 10

with codes 13, 17, 96, 121, 184, 196, 284, 309, 446, 459, 546, 571, and 582, respectively. This breadth-first property means that as the list grows, we get a better approximation of the image, that is, successive nodes in the list lead to a better approximation. Finally, increasing the resolution of the image does not require extensive recoding of the codes for the existing nodes.

Our locational codes differ from those used by Gargantini [5] and Abel and Smith [1]. They make use of fixed length codes, that is, for a $2^n \times 2^n$ image, all nodes are encoded by a sequence of n base 5 digits where one of the five values designates a do not care condition. It is used in the codes of nodes corresponding to blocks that are larger than one pixel. The coding sequence is defined as follows. Each node in the quadtree is represented by an n element sequence $\langle q_i \rangle = \langle q_{n-1}, \ldots, q_1, q_0 \rangle$ constructed from the digits {0, 1, 2, 3, 4}. Once again, let $\langle x_i \rangle$ represent the path of nodes from the root of the quadtree to x_m, the desired node, such that x_n = root of the quadtree and x_i = FATHER(x_{i-1}). The directions NW, NE, SW, and SE are represented by directional codes 0, 1, 2, and 3, respectively, and are accessed by the function SONTYPE4. 4 corresponds to a do not care condition.

$$q_i = \begin{cases} 4 & 0 \le i < m \\ \text{SONTYPE4}(x_i) & m \le i < n. \end{cases}$$

For example, node 10 of Figure 1 would be encoded by the number $q = 138$. It can be decoded into the sequence of directional codes $\langle q_i \rangle = \langle 1, 0, 2, 3 \rangle$, that is, $q = 1 \cdot 5^3 + 0 \cdot 5^2 + 2 \cdot 5^1 + 3 \cdot 5^0$.

This encoding, termed a q_i sequence, has the interesting property that when the codes of the BLACK nodes are sorted in increasing order, the resulting sequence is the postorder traversal of the BLACK portion of the tree. A comparison of q_i with the z_i sequence described

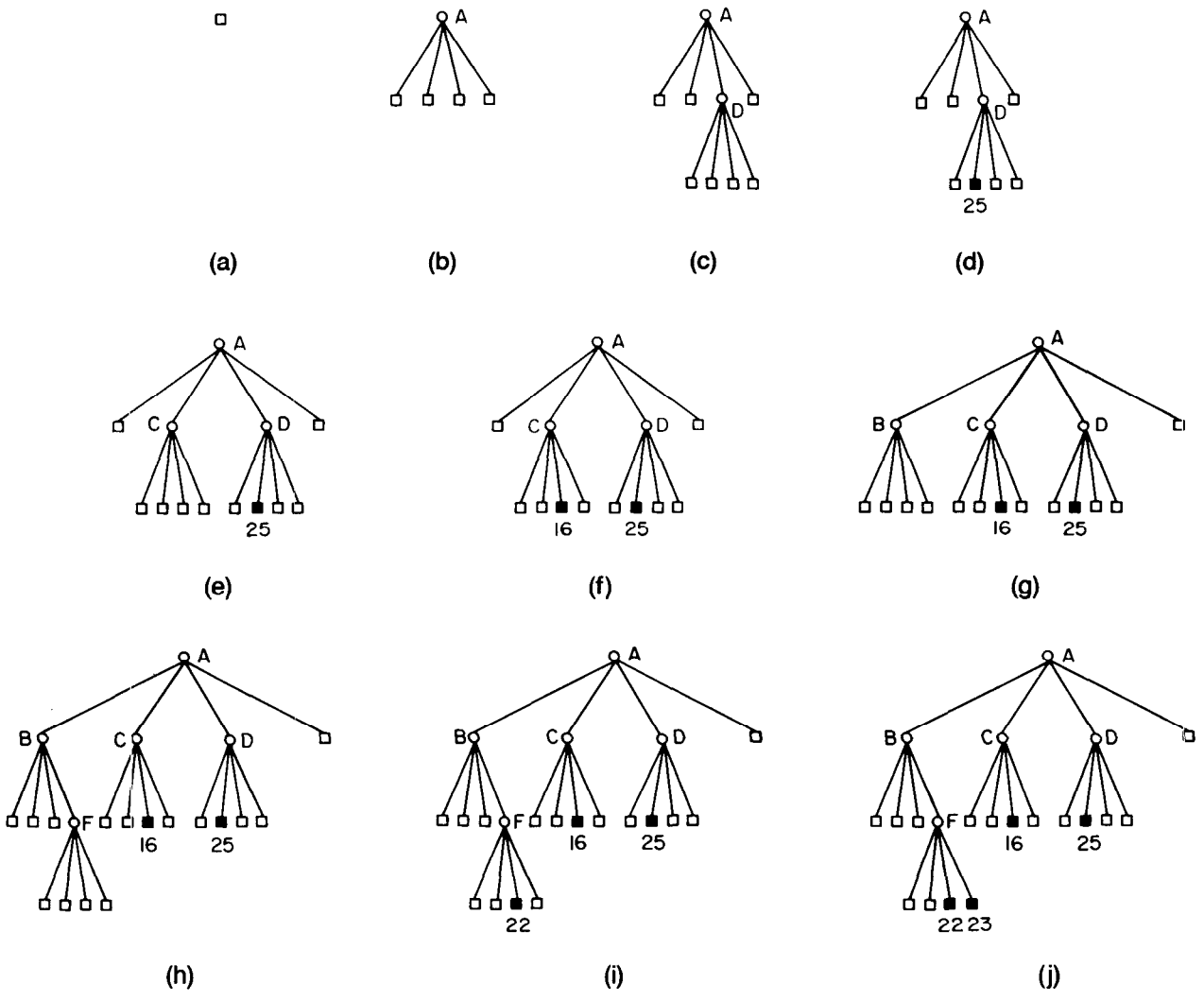


FIGURE 2. The Steps in Adding Nodes 25, 16, 22, and 23 When Constructing the Quadtree Corresponding to Figure 1

earlier reveals a number of shortcomings. First, using q_i it is more complex to decode a locational code to yield the path from the root of a quadtree to the root of the subquadtree. Second, increasing the resolution of the image requires recoding q_i for the existing nodes. In particular, their codes must be multiplied by 5 to a power equal to the increase in resolution. Finally, q_i does not have the progressive approximation property of z_i.

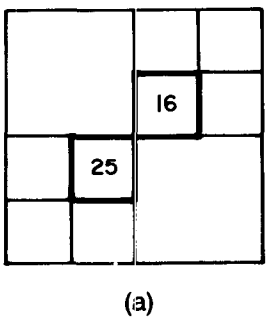
Using a q_i or z_i sequence encoding for the BLACK nodes of the quadtree, the tree representation (i.e., with pointers) is quite easy to obtain. For example, procedures BUILD_TREE and ADDNODE, given below, enable reconstruction of the tree representation from z_i. To facilitate the explanation of the algorithm, we refer to Figure 1 as an example and build its tree for the sequence of nodes 25, 16, 22, 23, 28, 14, 27, 29, 20, 38, 19, 21, and 10 in this order. Figures 2a–j show the intermediate trees obtained during the process of adding nodes 25, 16, 22, and 23. BUILD_TREE has as its input a list of codes for the BLACK nodes and invokes ADDNODE to add each element thereof to the tree. Initially, the image is assumed to be WHITE and thus the root is initialized to be of type WHITE (e.g., node *A* in Figure 2a). ADDNODE decodes the code for the node (i.e., the path to the node from the root of the tree) and in the process may need to expand a node (e.g., the transition from Figure 2a to Figures 2b and 2c when adding node 25). In such a case, the node's type is changed to GRAY and its four sons take on its previous value (e.g., the color of node *A* is changed to GRAY in the transition from Figure 2a to Figure 2b). When the decoding process is done, the node's type is complemented (e.g., node 25 becomes BLACK in the transition from Figure 2c to Figure 2d). Applying procedures BUILD_TREE and ADDNODE to the remaining nodes (i.e., nodes 28, 14, 27, 29, 20, 38, 19, 21, and 10) yield Figure 1.

```
node procedure BUILD_TREE(L);
/* Construct a quadtree from the list L of nodes in
   terms of locational codes. L is a pointer to a list
   implemented as a record of type nlist having two
   fields DATA and NEXT corresponding to the data
   stored in the list and a pointer to the next element
   in the list. */
begin
  value pointer nlist L;
  pointer node ROOT;
  ROOT ← create(node);  /* Assume that the initial
                           image is not empty */
  NODETYPE(ROOT) ← 'WHITE';
  while not empty(L) do
    begin
      ADDNODE(ROOT,DATA(L));
      L ← NEXT(L);
    end;
  return(ROOT);
end;
```

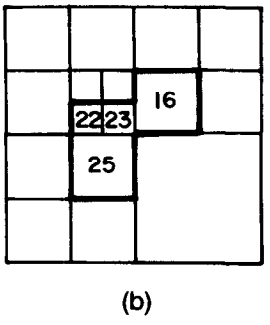
```
procedure ADDNODE(ROOT,B);
/* Add a node with locational code B to the quadtree
   rooted at node ROOT. */
begin
  value pointer node ROOT;
  value integer B;
  quadrant Q;
  while (B mod 5) ≠ 0 do
    begin
      if not GRAY(ROOT) then
        begin  /* Expand a terminal node */
          for I in {'NW', 'NE', 'SW', 'SE'} do
            begin
              SON(ROOT,I) ← create(node);
              NODETYPE(SON(ROOT,I))
                  ← NODETYPE(ROOT);
            end;
          NODETYPE(ROOT) ← GRAY;
        end;
      ROOT ← SON(ROOT,DIR(B mod 5));
      /* DIR converts a directional code to a
         quadrant */
      B ← B div 5;
    end;
  NODETYPE(ROOT)
      ← COMPLEMENT(NODETYPE(ROOT));
  /* The node's type is opposite to the type of the node
     that was being added or expanded, that is, a
     WHITE(BLACK) node is being replaced by a
     BLACK(WHITE) node. */
end;
```

3. HIERARCHICAL APPROXIMATION METHODS

By virtue of its hierarchical structure the quadtree lends itself to serve as an image approximation device. By truncating the tree (i.e., ignoring all nodes below a certain level), we get a crude approximation. Ranade, Rosenfeld, and Samet [18] define two basic variants termed an inner and outer approximation. Given an image I, the inner approximation IB(k) is the binary image defined by the BLACK nodes at levels $\geq k$. Figures 3a and 3b (p. 978) show IB(2) and IB(1), respectively, for Figure 1. The outer approximation, OB(k), is the binary image defined by the BLACK nodes at levels $\geq k$ and the GRAY nodes at level k. Figures 4a and 4b (p. 978) show OB(2) and OB(1), respectively, for Figure 1. At this point, let us use $\subseteq$ and $\supseteq$ to indicate set inclusion in the sense that $A \subseteq B$ and $B \supseteq A$ imply that the space spanned by A is a subset of the space spanned by B. It can be shown that $\text{IB}(n) \subseteq \text{IB}(n-1) \subseteq \cdots \subseteq \text{IB}(0) = I$ and $\text{OB}(n) \supseteq \text{OB}(n-1) \supseteq \cdots \supseteq \text{OB}(0) = I$. Alternatively, we can approximate the image by using its complement, $\bar{I}$. that is, the WHITE blocks. We define IW(k) and OW(k) in an analogous manner to that of IB(k) and OB(k), respectively, except in terms of WHITE blocks. It can be shown that $\text{IW}(n) \subseteq \text{IW}(n-1) \subseteq \cdots \subseteq \text{IW}(0) = \bar{I}$ and $\text{OW}(n) \supseteq \text{OW}(n-1) \supseteq \cdots \supseteq \text{OW}(0) = \bar{I}$. Moreover, Ranade, Rosenfeld, and Samet [18] show that

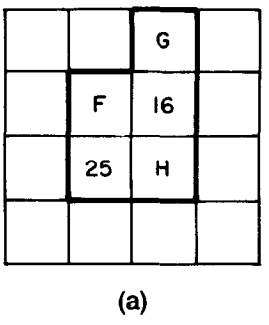


(a)

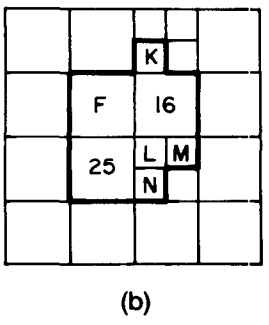


(b)

FIGURE 3. (a) IB(2) and (b) IB(1) for Figure 1



(a)



(b)

FIGURE 4. (a) OB(2) and (b) OB(1) for Figure 1

the outer approximations to I are actually the inner approximations to $\bar{I}$ = that is, $OB(k) = \overline{IW(k)}$. Similarly, it can be shown that the inner approximations to I are actually the outer approximations to $\bar{I}$, that is, $IB(k) = \overline{OW(k)}$.

Sloan and Tanimoto [26] treat the problem of transmitting an image by successively approximating it by use of pyramid-based approaches [27]. They are able to handle gray scale images. They propose a number of methods. However, none feature any compression. Knowlton [14] addresses a similar problem. He makes use of a binary tree version of a quadtree. In essence, an image is split into two halves alternating between horizontal and vertical splits. Much of the compression is obtained by using special coding techniques to encode primitive 2×3 blocks. The methods we discuss in the following section make no use of such techniques.

4. FOREST-BASED APPROXIMATION METHODS

Jones and Iyengar [10] introduced the concept of a forest of quadtrees which is a decomposition of a quadtree into a collection of subquadtrees, each of which corresponds to a maximal square. The maximal squares are identified by refining the concept of a nonterminal node to indicate some information about its subtrees. An internal node is said to be of type GB if at least two of its sons are BLACK or of type GB. Otherwise, the node is said to be of type GW. For example, in Figure 1, nodes *F*, *J*, and *M* are of type GB and nodes *A*, *B*, *C*, *D*, *E*, *G*, *H*, *I*, *K*, *L*, and *N* are of type GW. Each BLACK node or an internal node with a label GB is said to be maximal square. A *BLACK forest* is the minimal set of maximal squares that are not contained in other maximal squares and that span the BLACK area of the image. Thus, the BLACK forest corresponding to Figure 1 is {*F*, 10, 16, 25, 27, *M*, 38} and their corresponding subtrees. The elements of the BLACK forest are specified by locational codes (although Jones and Iyengar use a different definition than z_i). Such a representation can lead to a savings of space since large WHITE areas are ignored by it.

A forest can also be used as an approximation where we treat its elements as BLACK and all remaining nodes as WHITE. In the remainder of this section, we expand on the use of a forest to approximate images. It is useful to sort the nodes of the forest according to their codes. We use the locational codes given by z_i. For example, for Figure 1, the nodes will appear in the order 25, 16, *F*, *M*, 27, 38, and 10. This order is a partial ordering $(S, \geq)$ such that $S_i \geq S_{i+1}$ means that the block subsumed by S_i is $\geq$ in size than the block subsumed by S_{i+1}. In fact, for a breadth-first traversal we only need to process the nodes in an order that satisfies the above subsumption relation. It should be clear that a sorted list is just one of many possible orderings satisfying the subsumption relation.

The BLACK forest approximation can be defined as follows. Let FB(*P*) be the quadtree constructed by coloring BLACK the roots of the subquadtrees comprising the forest of the quadtree rooted at node *P*. FB(*P*) is empty when *P* is a WHITE terminal node and likewise it is equal to *P* when *P* is a BLACK terminal node. FB(*P*) can be further refined to more closely mirror the concept of an approximation. Let FB(*P*, *k*), the *k*th order approximation to FB(*P*), be the nodes at levels $\geq k$ of FB(*P*). For example, in Figure 1 we have FB(*A*, 2) = {25, 16, *F*}; FB(*A*, 1) = {25, 16, *F*, *M*}; FB(*A*, 0) = {25, 16, *F*, *M*, 27, 38, 10} = FB(*A*). FB(*A*, 4) = FB(*A*, 3) = { }. Figures 5a, 5b, and 5c show FB(*A*, 2), FB(*A*, 1), and FB(*A*, 0), respectively. Clearly, $FB(P, n) \subseteq FB(P, n-1) \subseteq \cdots \subseteq FB(P, 0) = FB(P)$.

Note that any node can be approximated by a BLACK forest of quadtrees, that is, not just the root of the quadtree corresponding to the image. Thus, we can approximate an image by a sequence of BLACK forests of quadtrees where the sequence is defined by replacing nonterminal node components of a BLACK forest by their BLACK forests. More formally, let FBB(*i*), $0 \leq i \leq n$, be defined as follows:

$$\text{FBB}(i) = \begin{cases} \{\text{FB(root, 0)}\} & i = n \\ \{\text{FB}(j, 0) \mid j \in \text{FBB}(i+1) \textbf{ and not}(\text{GRAY}(j))\} \\ \quad \cup \{\text{FB}(k, 0) \mid j \in \text{FBB}(i+1) \\ \qquad \textbf{and } \text{GRAY}(j) \textbf{ and } \text{FATHER}(k) = j\} \\ & 0 \leq i < n. \end{cases}$$

For example, in Figure 1 we have FBB(4) = {25, 16, *F*, *M*, 27, 38, 10}. FBB(3) is obtained by adding the BLACK

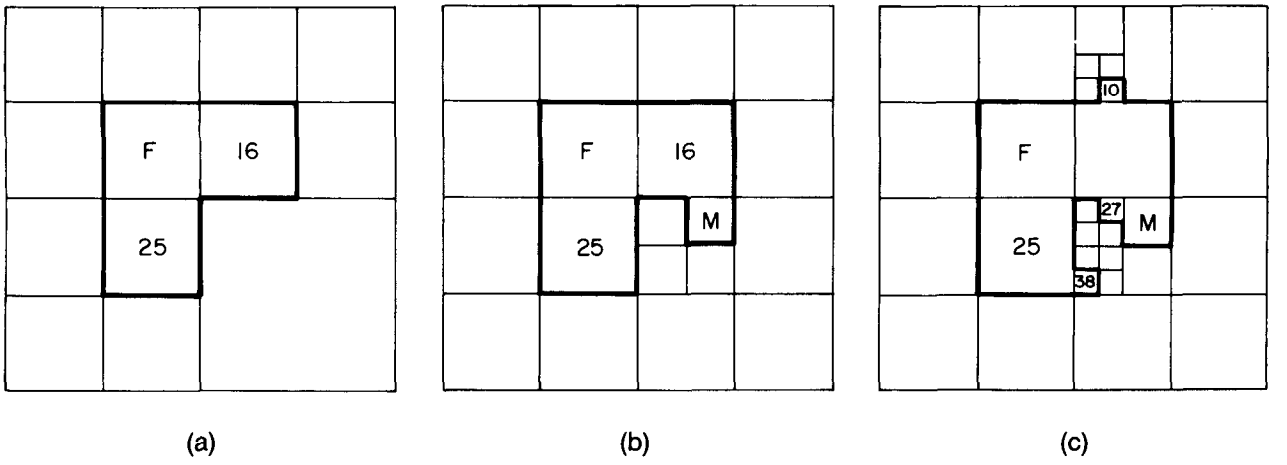


FIGURE 5. (a) FB(*A*, 2), (b) FB(*A*, 1), and (c) FB(*A*, 0) for Figure 1

forests of nodes F (i.e., nodes 19, J, 22, and 23) and M (i.e., nodes 28 and 29) to yield FBB(3) = { 25, 16, J, 22, 23, 28, 27, 29, 38, 19, 10}. FBB(2) is obtained by adding the BLACK forests of node J (i.e., nodes 14, 20, and 21) to yield FBB(2) = {25, 16, 22, 23, 20, 14, 27, 29, 28, 38, 19, 21, 10}. No additional nodes are added by FBB(1) and FBB(0), that is, FBB(2) = FBB(1) = FBB(0) = I. Figures 6a and 6b show FBB(4) and FBB(3), respectively. Clearly, FBB(n) $\supseteq$ FBB($n-1$) $\supseteq \cdots \supseteq$ FBB(0) = I.

Approximation FBB provides a closer approximation to the image than OB in the sense that OB(i) $\supseteq$ FBB(i) for all i. This is because the BLACK forest approximation (FBB) only includes GRAY nodes at level i if they represent $2^i \times 2^i$ blocks that are at least $1/2^i$ BLACK and OB includes GRAY nodes at level i if any fraction of their corresponding $2^i \times 2^i$ block is BLACK. Actually, the index i in FBB(i) corresponds to an iteration whereas the index i in OB(i) corresponds to a level. Nevertheless, OB(i) $\supseteq$ FBB(i) because both OB(i) and FBB(i) contain all terminal nodes at levels $k \geq i$, and all nodes of FBB(i) at levels $j < i$ are contained in the GRAY nodes at level i which are part of OB(i).

We can also have a forest approximation that is made up entirely of nodes corresponding to WHITE blocks. In other words, we are approximating the complement of the image $\bar{I}$. Such a forest is defined analogously to the one presented earlier using the BLACK blocks, that is, each quadtree is a collection of subquadtrees, each of which corresponds to a maximal square. A *WHITE forest* is the minimal set of maximal squares that are not contained in other maximal squares and that spans the WHITE area of the image. Now, all that remains is to define a maximal square. As in the case of BLACK forests, maximal squares are defined in terms of inter-

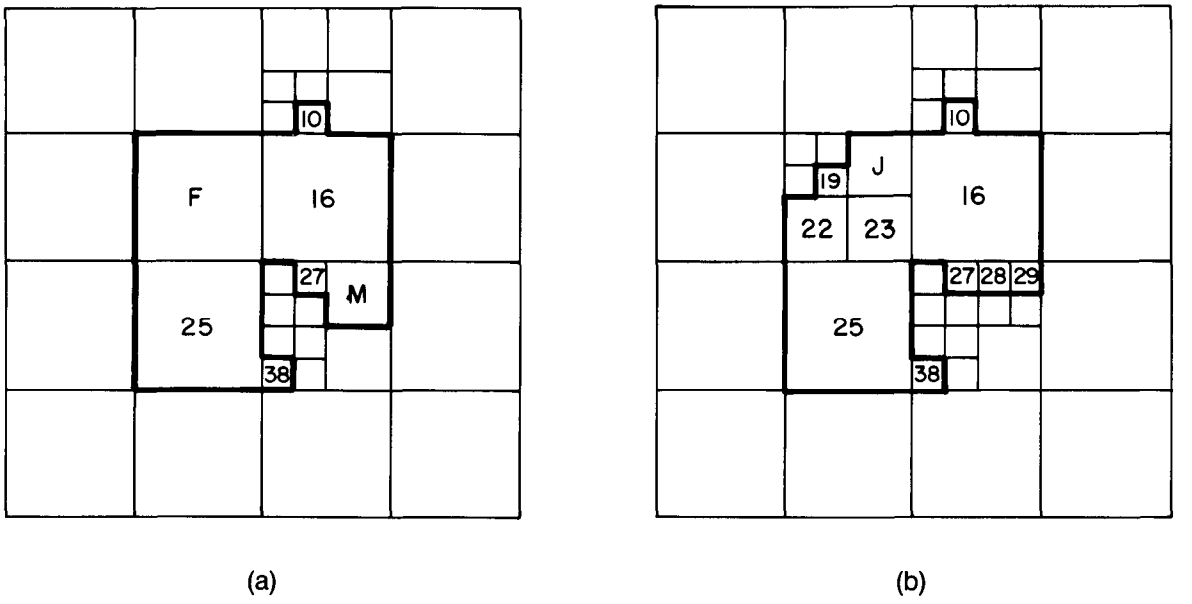


FIGURE 6. (a) FBB(4) and (b) FBB(3) for Figure 1

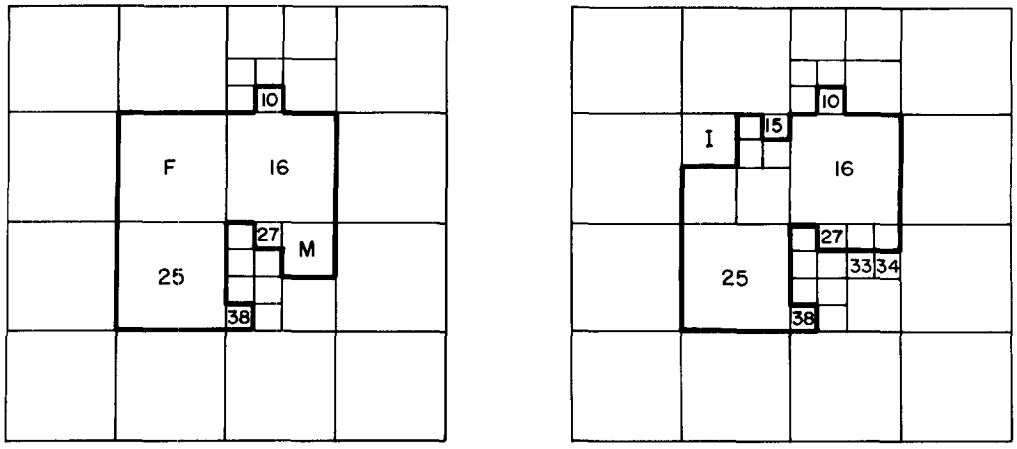


FIGURE 7. (a) FBW(4) and (b) FBW(3) for Figure 1

nal (i.e., GRAY) nodes. There are two ways in which we can proceed. First, we can retain our definitions of internal nodes in terms of GW and GB. Alternatively, we can define an internal node to be of type GW′ if at least two of its sons are WHITE or of type GW′. Otherwise, the node is said to be of type GB′. We choose to retain the original definition because it complements the way the nodes were defined to build the BLACK forest. Thus, given a tree labeled with GB and GW, both BLACK and WHITE forests may be extracted; whereas when using GW′ and GB′, prior to extracting the WHITE forest from a tree labeled with GB and GW, the tree must be relabeled. We use the following notation in our discussion of WHITE forests. FW(P) is the WHITE forest corresponding to the quadtree rooted at node P. FW(P) is further refined in terms of FW(P, k), the kth *order approximation to* FW(P), *as the nodes at levels* $\geq k$ of FW(P). For example, in Figure 1, we have FW(A, 4) = {A} and, in fact, FW(A, 3) = FW(A, 2) = FW(A, 1) = FW(A, 0) = FW(A) = {A}; FW(F, 2) = { }; FW(F, 1) = {I}; FW(F, 0) = {I, 15}; FW(H, 2) = {H}; FW(H, 1) = {H}; FW(H, 0) = {H}. Clearly, FW(P, n) $\subseteq$ FW(P, $n-1$) $\subseteq \cdots \subseteq$ FW(P, 1) = FW(P).

Similarly, we can approximate the complement of the image by a sequence of forests of WHITE quadtrees where the sequence is defined by replacing nonterminal node components by their WHITE forests. More formally, let FWW(i), $0 \leq i \leq n$, be defined as follows:

$$\text{FWW}(i) = \begin{cases} \{\text{FW(root, 0)}\} & i = n \\ \{\text{FW}(j, 0) \mid j \in \text{FWW}(i+1) \\ \quad \textbf{and not}(\text{GRAY}(j))\} \\ \quad \cup \{\text{FW}(k, 0) \mid j \in \text{FWW}(i+1) \\ \quad \textbf{and } \text{GRAY}(j) \\ \quad \textbf{and } \text{FATHER}(k) = j\} & 0 \leq i < n. \end{cases}$$

For example, in Figure 1 we have FWW(4) = {A}. FWW(3) is obtained by adding the WHITE forests of node A (i.e., nodes B, C, D, and E) to yield FWW(3) = {B, C, D, E}. FWW(2) is obtained by adding the WHITE forests of node B (i.e., nodes 1, 2, 11, I, and 15), C (i.e, nodes G, 5, and 17), D (i.e., nodes 24, 40, and 41), and E (i.e., nodes H, 30, 42, and 43) to yield FWW(2) = {1, 2, 11, I, 15, G, 5, 17, 24, 40, 41, H, 30, 42, 43}. Continuing this procedure we find FWW(1) = {1, 2, 11, 12, 13, 18, 15, 3, 4, K, 8, 5, 17, 24, 40, 41, L, 33, 34, N, 37, 30, 42, 43} and FWW(0) = {1, 2, 11, 12, 13, 18, 15, 3, 4, 6, 7, 9, 8, 5, 17, 24, 40, 41, 26, 31, 32, 33, 34, 35, 36, 39, 37, 30, 42, 43} = $\overline{I}$. Clearly, FWW(n) $\supseteq$ FWW($n-1$) $\supseteq \cdots \supseteq$ FWW(0) = IW(0) = $\overline{I}$.

Earlier we saw that OB(i) $\supseteq$ FBB(i) for all i and we have an analogous relationship between OW(i) and FWW(i), that is, OW(i) $\supseteq$ FWW(i) and hence FWW provides a closer approximation to the inverse image than OW. We can also make use of FWW to approximate I by working with its complement, $\overline{\text{FWW}}$, defined as follows:

$$\overline{\text{FWW}(i)} = \begin{cases} \overline{\{\text{FW(root, 0)}\}} & i = n \\ \overline{\{\text{FW}(j, 0) \mid j \in \text{FWW}(i+1) \\ \quad \textbf{and not}(\text{GRAY}(j))\}} \\ \quad \cup \overline{\{\text{FW}(k, 0) \mid j \in \text{FWW}(i+1) \\ \quad \textbf{and } \text{GRAY}(j) \\ \quad \textbf{and } \text{FATHER}(k) = j\}} & 0 \leq i < n. \end{cases}$$

Clearly, $\overline{\text{FWW}(i)} \supseteq \overline{\text{OW}(i)}$ for all i. Recalling that $\overline{\text{OW}(i)}$ = IB(i), we have IB(i) $\subseteq \overline{\text{FWW}(i)}$. Also, FWW($i$) $\supseteq \overline{I}$ implies that $\overline{\text{FWW}(i)} \subseteq I$. In fact, we have just shown the existence of better approximations to I (i.e., FBB and FWW) than OB and IB and in the process have proven the following theorem.

THEOREM 1.
IB(i) $\subseteq \overline{\text{FWW}(i)} \subseteq I \subseteq$ FBB(i) $\subseteq$ OB(i) *for all* i $\quad 0 \leq i \leq n$.

Use of approximation FBB results in overestimating the area spanned by the image while use of $\overline{\text{FWW}}$ results in underestimating the area. In essence, we are approximating the image solely by use of BLACK blocks or solely by use of WHITE blocks. However, we could

also approximate the image by a combination of BLACK and WHITE blocks. What we do is use FBB(n) for the first level of approximation; augment all elements of FBB(n) which correspond to GRAY nodes in the original quadtree by use of FWW; and repeat the alternating process until no GRAY nodes are left. More formally, we define a sequence FBW(i) as follows:

$$\text{FBW}(i) = \begin{cases} \text{empty} & i = n+1 \\ \text{FBB}(n) & i = n \\ \text{FBW}(i+1) \cup \{\text{FW}(j) \mid j \in \text{FBW}(i+1) & \\ \qquad \textbf{and } j \notin \text{FBW}(i+2)\} & \\ \qquad (n-i) \textbf{ mod } 2 = 1 \textbf{ and } i < n & \\ \text{FBW}(i+1) \cup \{\text{FB}(j) \mid j \in \text{FBW}(i+1) & \\ \qquad \textbf{and } j \notin \text{FBW}(i+2)\} & \\ \qquad (n-i) \textbf{ mod } 2 = 0 \textbf{ and } i < n. & \end{cases}$$

For example, in Figure 1 we have FBW(4) = {25, 16, *F*, *M*, 27, 38, 10}. FBW(3) is obtained by adding the WHITE forests of node *F* (i.e., nodes *I* and 15) and *M* (i.e., nodes 33 and 34) to yield FBW(3) = {25, 16, *F*, *M*, 27, 38, 10, *I*, 15, 33, 34}. FBW(2) is obtained by adding the BLACK forests of node *I* (i.e., node 19) to yield FBW(2) = {25, 16, *F*, *M*, 27, 38, 10, *I*, 15, 33, 34, 19} = FBW(1) = FBW(0) = *I*. Figures 7a and 7b show FBW(4) and FBW(3), respectively. Note that in Section 5 we discuss a symmetric approximation which makes use of FWW.

Procedure ENCODE_FBW, given below, generates the FBW(0) encoding of an image. It does this by successively generating in order, FBW(n), FBW($n-1$), ... until FBW(i) is encountered such that no more GRAY nodes need to be expanded. In essence, as FBW(n) (i.e., a BLACK forest) is generated, all of its elements that correspond to GRAY nodes are output as well as placed on a list termed WLIST (e.g., nodes *F* and *M* in Figure 1). Next, we replace all elements of WLIST by their WHITE forests and thereby generate FBW($n-1$). During this process all elements of FBW($n-1$) that correspond to GRAY nodes are output as well as placed on the BLIST (e.g., node *I*) to be used in the generation of FBW($n-2$). The process terminates when encountering an empty BLIST or WLIST. Note that WLIST is initially empty and BLIST is initialized to the entire image (e.g., node *A* in Figure 1). ENCODE_FBW makes use of procedures FOREST_BLACK2 and FOREST_WHITE2 to generate the BLACK and WHITE forests corresponding to elements of BLIST and WLIST, respectively. Note that in addition to outputting a forest of the appropriate color, they also construct a list of the nonterminal nodes of the forest to serve as input for the next level of encoding.

ENCODE_FBW yields the nodes for FBW(0) in the order of FBW(n), FBW($n-1$), FBW($n-2$), ..., FBW(0). This is useful because it means that the reverse process of building a quadtree from the transmitted codes results in obtaining successive approximations to the image. In fact, procedure BUILD_TREE, given earlier in Section 2, to construct a quadtree given the set of locational codes for all of its BLACK blocks can be used to construct the quadtree from the FBW approximation. This is true as long as the codes for the nodes are processed (i.e., added to the tree) in an order so that for any two nodes *P* and *Q* such that *P* is an ancestor of *Q*, *P* is added to the tree before *Q*. It should be clear that procedure ENCODE_FBW yields the nodes in such an order. In fact, as will become apparent from the discussion of BUILD_TREE below, if we do not have such an order, then we would have to specify the colors corresponding to the elements of FBW (i.e., BLACK, WHITE, GB, or GW). Note that this ordering property can also be satisfied by a sorted sequence, a breadth-first traversal, and even a preorder traversal. In the subsequent discussion, we use the term *transmit* to denote the encoding process.

```
procedure ENCODE_FBW(P);
/* Given a 2^N × 2^N image represented by a quadtree
   such that P points to its root, construct its FBW
   approximation. The approximations are output in
   the order FBW(N), FBW(N − 1), ..., FBW(0). For
   each approximation FBW(i), all nodes at level k are
   output before nodes of level j where j < k. The
   approximations alternate between BLACK forests
   and WHITE forests with the first approximation
   being a BLACK forest. BLIST is a list of nodes for
   whom a BLACK forest is to be constructed and
   WLIST is a list of nodes for whom a WHITE forest is
   to be constructed. BLIST and WLIST are both of
   type list which is a record having four fields PTR,
   PATH, LEV, NEXT corresponding respectively to a
   pointer to a node, the locational code for the path
   from the root of the quadtree to the node, N minus
   the level of the node, and a pointer to the next
   element in the list. BLIST is initially set to P. */
begin
  value pointer node P;
  pointer list BLIST, WLIST;
  global integer N;
  WLIST ← "empty";
  /* Initialize BLIST to P with locational code 0 and
     level N: */
  ADDTOLIST(BLIST,P,0,0);
  while true do
    begin
      if empty(BLIST) then return;
      while not empty(BLIST) do
        begin
          FOREST_BLACK2(PTR(BLIST),
              PATH(BLIST), LEV(BLIST), WLIST);
          BLIST ← NEXT(BLIST);
        end;
      if empty(WLIST) then return;
      while not empty(WLIST) do
        begin
          FOREST_WHITE2(PTR(WLIST),
              PATH(WLIST), LEV(WLIST), BLIST);
          WLIST ← NEXT(WLIST);
        end;
    end;
end;
```

```
procedure FOREST_BLACK2(P,B,I,WLIST);
/* Output the BLACK forest nodes for the quadtree
   rooted at P with locational code B and at level N - I
   given a 2^N x 2^N image. All elements of the BLACK
   forest which are nonterminal nodes are also added
   to the list WLIST for subsequent expansion into a
   WHITE forest. */
begin
  value pointer node P;
  value integer B, I;
  reference pointer node WLIST;
  global integer N;
  quadrant Q;
  if GB(P) then
    begin
      output(B);
      ADDTOLIST(WLIST,P,B,I);
    end
  else if BLACK(P) then output(B)
  else if GW(P) then
    begin
      for Q in {'NW', 'NE', 'SW', 'SE'} do
        FOREST_BLACK2(SON(P,Q),
            B+QCODE(Q)*5**I, I+1,WLIST);
        /* QCODE converts a quadrant to a locational
           code */
    end;
end;


procedure FOREST_WHITE2(P,B,I,BLIST);
/* Output the WHITE forest for the quadtree rooted at
   P with locational code B and at level N - I given a
   2^N x 2^N image. All elements of the WHITE forest
   which are nonterminal nodes are also added to the
   list BLIST for subsequent expansion into a BLACK
   forest. */
begin
  value pointer node P;
  value integer B, I;
  reference pointer node BLIST;
  global integer N;
  quadrant Q;
  if GW(P) then
    begin
      output(B);
      ADDTOLIST(BLIST,P,B,I);
    end
  else if WHITE(P) then output(B)
  else if GB(P) then
    begin
      for Q in {'NW', 'NE', 'SW', 'SE'} do
        FOREST_WHITE2(SON(P,Q),
            B + QCODE(Q)*5**I, I+1,BLIST);
    end;
end;
```

At this point, let us examine the mechanics of BUILD_TREE more closely to see how and why it works correctly. Recall that whenever a node is added, a path (designated by the node's locational code) is traced from the root of the quadtree to the node. In the process, nodes may need to be expanded (e.g., the transition from Figure 2a to Figures 2b and 2c when adding node 25). In such a case, the node's type is changed to GRAY and its four sons take on its previous value (e.g., node *A* in the transition from Figure 2a to Figure 2b). When the decoding process is done, the node's type is complemented (e.g., node 25 changes from WHITE to BLACK in the transition from Figure 2c to Figure 2d). In the case of the FBW sequence, replacement of a BLACK (WHITE) node by its WHITE (BLACK) forest proceeds in the same way. For example, Figure 8a shows the result of applying BUILD_TREE to FBW(4) = {25, 16, *F*, *M*, 27, 38, 10} of Figure 1. Next, we process FBW(3) which means that we must add nodes *I*, 15, 33, and 34. Figures 8b–8h illustrate this process. The presence of node *I* implies that BLACK node *F* must be replaced by its WHITE forest. This is accomplished in two steps. First, node *F* is changed to a GRAY node and gains four BLACK sons (Figure 8b). Next, node *I* is labeled and in the process is changed from BLACK to WHITE (Figure 8c). Figures 8d and 8e show how node 15 is added while Figures 8f–8h show the process of adding nodes 33 and 34. The final step is to process FBW(2) which means that we must add node 19. In this case, the presence of node 19 implies that WHITE node *I* must be replaced by its BLACK forest. Figures 8i and 8j illustrate this process.

Approximation FBW satisfies the following relationships FBW(n) $\supseteq$ FBW($n-1$), FBW($n-1$) $\subseteq$ FBW($n-2$), FBW($n-2$) $\supseteq$ FBW($n-3$), and, in general, FBW($n-2i$) $\supseteq$ FBW($n-2i-1$) and FBW($n-2i-1$) $\subseteq$ FBW($n-2i-2$). Furthermore, it is easy to show that FBW(n) $\supseteq$ FBW($n-2$) $\supseteq \cdots \supseteq$ FBW($n-2i$) $\supseteq \cdots \supseteq$ FBW(0) = $I \supseteq \cdots \supseteq$ FBW($n-2i-1$) $\supseteq \cdots \supseteq$ FBW($n-3$) $\supseteq$ FBW($n-1$). In other words, the approximations FBW spiral in from both sides of I in converging to I. Note that individually, FBB and $\overline{\text{FWW}}$ may, at times, be better approximations to I than FBW—that is, there exist images for which the amount of BLACK by which FBB overestimates I is less than the amount of BLACK by which FBW underestimates I. As an example, suppose that WHITE node 18 in Figure 1 has been replaced by GRAY node *P* and four sons 44, 45, 46, and 47 where nodes 44, 45, and 46 are WHITE and node 47 is BLACK (see Figure 9 where the subtree rooted at node *F* is shown). In such a case FBB(3) = {25, 16, *J*, 22, 23, 28, 27, 29, 38, 19, 10, 47}, FBW(3) = {25, 16, *F*, *M*, 27, 38, 10, *I*, 15, 33, 34}, and FBB(3) overestimates the BLACK area spanned by node *F* by 4 pixels while FBW(3) underestimates the same BLACK area by 5 pixels. Of course, an image can also be constructed where the opposite is true—that is, the amount of BLACK by which FBB overestimates I is greater than the amount of BLACK by which FBW underestimates I. As an example, suppose that BLACK node 20 in Figure 1 has been changed to a WHITE node. In this case, FBB(3) = {25, 16, *J*, 22, 23, 28, 27, 29, 38, 19, 10}, FBW(3) = {25, 16, *F*, *M*, 27, 38, 10, *I*, 15, 33, 34, 20}, and FBW(3) underestimates the BLACK

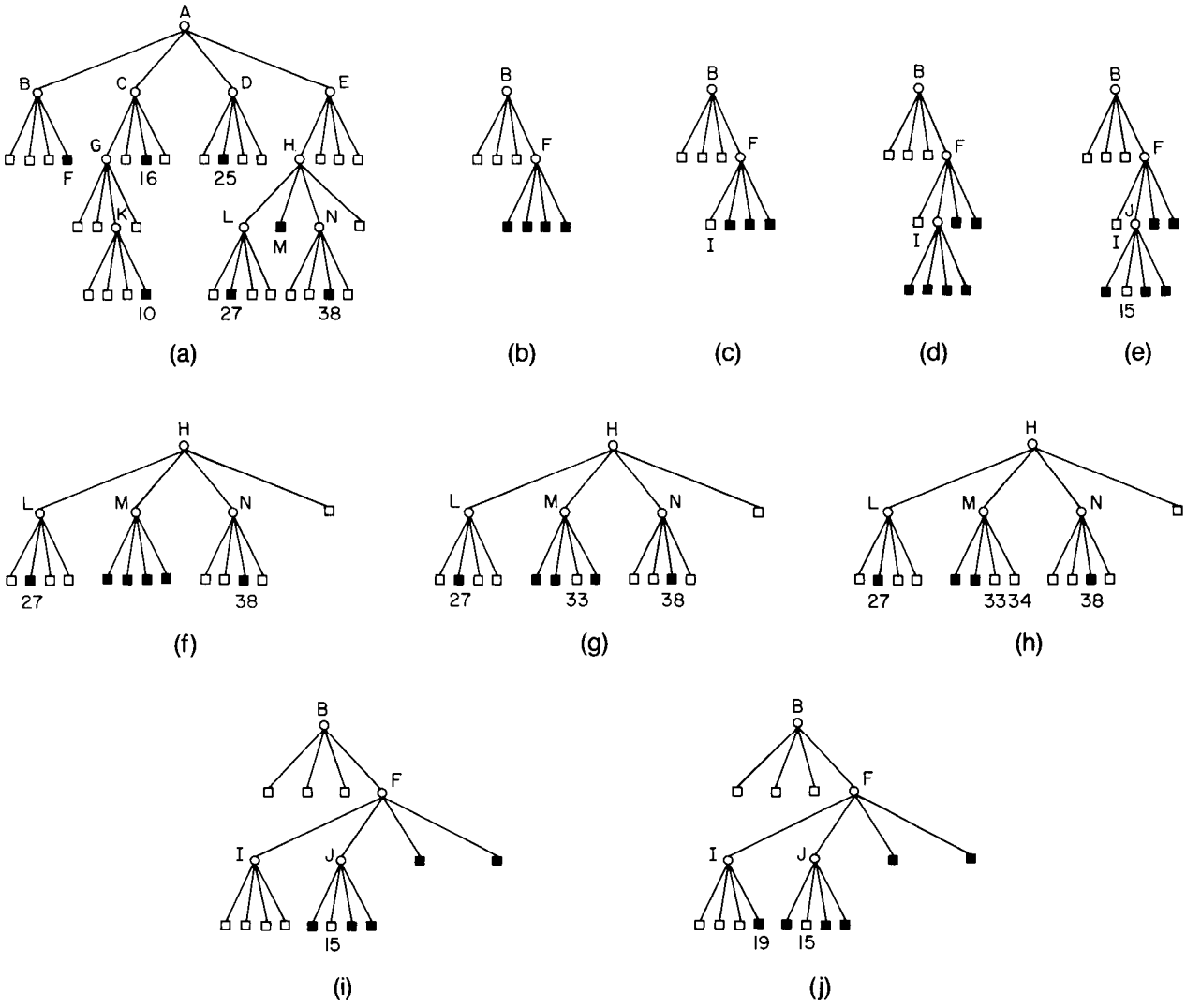


FIGURE 8. The Steps in Adding Nodes *I*, 15, 33, 34, and 19 When Reconstructing the Quadtree from the FBW Encoding of Figure 1

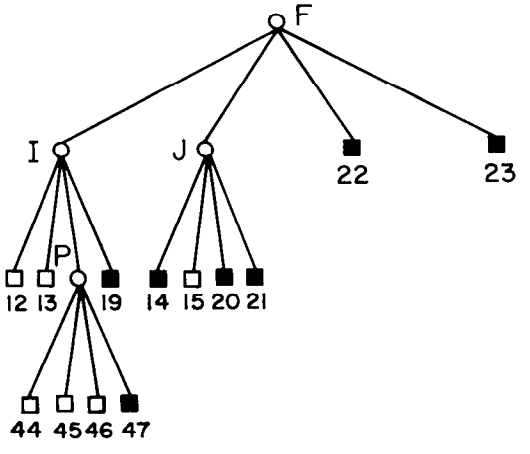


FIGURE 9. Modifications to Figure 1 Showing the Example Where FBB Overestimates the Image by Less Than What FBW Underestimates it

area spanned by node F by 1 pixel while FBB(3) overestimates the same area by 2 pixels.

FBW is attractive as an approximation method because it converges to the image from both sides (i.e., it alternately overestimates and underestimates the BLACK component). Thus, it strikes a balance between using all BLACK nodes or all WHITE nodes to approximate the image as is the case with FBB and $\overline{\text{FWW}}$. FBW, by definition, also has the property that FBW(j) $\subseteq$ FBW(i) for $0 \leq i \leq j \leq n$. In this case, by $\subseteq$ we mean that the nodes comprising approximation j are included in approximation i. Of course, FBB and $\overline{\text{FWW}}$ could also have been defined in an analogous manner by including GRAY nodes. Recall that FBW(0) = I and thus when we use procedure ENCODE_FBW to encode the image, we can get the successive approximations to I as the elements of FBW(0) are transmitted. Therefore, the first elements that are transmitted make up FBW(n) and the next terms lead to FBW($n - 1$), etc.

5. COMPRESSION

The FBW approximation has the interesting property that its use will often lead to compression in the sense that it reduces the amount of data that is needed to encode the image (and transmit it). Recall that we can represent a quadtree by merely specifying all of the BLACK blocks or all of the WHITE blocks. Depending on the image, we would use the color with the smaller cardinality in order to save storage. The FBW approximation consists of a combination of GRAY, BLACK, and WHITE nodes thereby striking a balance between using all BLACK or all WHITE. For example, encoding Figure 1 with FBW requires 12 nodes whereas the image contains 13 BLACK and 30 WHITE nodes.

Thus, aside from its superiority with respect to the quality of the resulting approximation, FBW also leads to compression. Let F, B, and W denote the number of nodes when encoding the quadtree using FBW, BLACK, and WHITE nodes, respectively. Compression is said to exist whenever $F < \text{MIN}(B, W)$. As we shall see below, variants of FBW can be constructed so that F is always $\leq \text{MIN}(B, W)$. Thus, we can guarantee that our approximation methods are always at least as good or better than encoding the quadtree by listing its BLACK nodes (or its WHITE nodes).

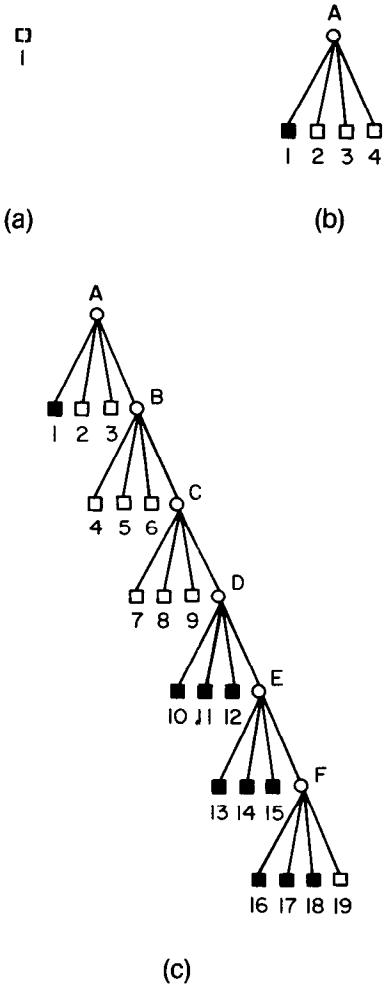


FIGURE 10. Examples Illustrating the Compression Factors Available Through the Use of FBW

To see the type of compression that is achievable, let $C = F/\text{MIN}(B, W)$ be a compression factor. C can be made as close to zero as desired. Figure 10a demonstrates the empty tree which has $F = 0$ (i.e., a WHITE node at the root) which we exclude. Figure 10b illustrates a tree with $F = 1$ but $C = 1$. Figure 10c is an example of the type of tree that yields much compression—that is, a small C value although it is by no means the minimum C for a $2^n \times 2^n$ image. In this case, $F = 3$ (nodes 1, D, and 19) while $B = 10$, $W = 9$, and $C = \frac{1}{3}$. For a $2^n \times 2^n$ image, a tree having depth $n = 2 \cdot m$ can be constructed such that $F = 3$ and $C = 3/(3 \cdot m) = 1/m$. Figure 10c is such a tree with $n = 6$. Note that such a tree has $3 \cdot m - 1$ WHITE nodes at levels $n - 1$ to $n/2$, 1 BLACK node at level $n - 1$, $3 \cdot (m - 1)$ BLACK nodes at levels $n/2 - 1$ to 1, 1 WHITE node at level 0, and 3 BLACK nodes at level 0. An upper bound on the number of nodes comprising FBW (i.e., F) is given by the following theorem whose proof appears in the Appendix.

THEOREM 2.
The maximum number of nodes in an FBW approximation is less than or equal to one plus the number of WHITE nodes in the quadtree (i.e., $F \leq W + 1$).

Figure 10c demonstrated how the lower bound on the compression factor can be approached. A checkerboard image is an example of the upper bound on F. Its FBW approximation has FBW(n) = {root} and FBWN($n - 1$) = {all terminal WHITE nodes}. Thus $F = W + 1 = B + 1$. Note that by Theorem 2 the upper bound on F is solely in terms of the WHITE nodes. This is better than the weaker upper bound of MAX(B, W) + 1 because $W \leq \text{MAX}(B, W)$.

The FBW approximation relies on alternating FBB and FWW approximations. We can also define an approximation, FWB, which alternates between FWW and FBB as follows:

$$\text{FWB}(i) = \begin{cases} \text{empty} & i = n + 1 \\ \text{FWW}(n) & i = n \\ \text{FWB}(i+1) \cup \{\text{FB}(j) \mid j \in \text{FWB}(i+1) \text{ and } j \notin \text{FWB}(i+2)\} & (n - i) \bmod 2 = 1 \text{ and } i < n \\ \text{FWB}(i+1) \cup \{\text{FW}(j) \mid j \in \text{FWB}(i+1) \text{ and } j \notin \text{FWB}(i+2)\} & (n - i) \bmod 2 = 0 \text{ and } i < n. \end{cases}$$

For example, in Figure 1 we have FWB(4) = {A}. FWB(3) is obtained by adding the BLACK forests of node A resulting in {A, 25, 16, F, M, 27, 38, 10}. FWB(2) is obtained by adding the WHITE forests of node F (i.e., nodes I and 15) and M (i.e., nodes 33 and 34) to yield FWB(2) = {A, 25, 16, F, M, 27, 38, 10, I, 15, 33, 34}. FWB(1) is obtained by adding the BLACK forests of node I (i.e., node 19) to yield FWB(1) = {A, 25, 16, F, M,

27, 38, 10, *I*, 15, 33, 34, 19} = FWB(0) = *I*. Of course, procedures ENCODE_FBW and BUILD_TREE must be slightly modified. In particular, we create a procedure ENCODE_FWB which differs from ENCODE_FBW only in that FOREST_WHITE2 is invoked before FOREST_BLACK2. The only necessary change to procedure BUILD_TREE is that initially the tree is BLACK rather than WHITE. Note that for our example FWB(3) = {root} + FBW(4), and, in general, as we shall see below, when the root of the tree is of type GB, {root} + FWB(i) = FBW($i - 1$), and when the root is of type GW, FWB($i - 1$) = {root} + FBW(i). We have the following theorem whose proof appears in the Appendix.

THEOREM 3.
The maximum number of nodes in an FWB approximation is less than or equal to the number of WHITE nodes in the quadtree (i.e., $F \leq W$).

As we see from Theorems 2 and 3 there really is not a big difference between using FBW and FWB. If we want to have the lowest upper bounds, then we can use FBW when the root is of type GW and FWB when the root is of type GB.

We can also obtain upper bounds in terms of *B*, the number of BLACK nodes. To do this, we redefine our approximation sequence. In particular, we relabel our quadtree with GB′ and GW′ as follows. An internal node is said to be of type GW′ if at least two of its sons are WHITE or of type GW′. Otherwise, the node is said to be of type GB′ (i.e., at least three of its sons are BLACK or of type GB′). For example, in Figure 1, nodes *A*, *B*, *C*, *D*, *E*, *G*, *H*, *I*, *K*, *L*, *M*, and *N* are of type GW′ and nodes *F* and *J* are of type GB′. We now redefine FB, FW, FBB, and FWW in terms of GB′ and GW′ to yield FB′, FW′, FBB′, and FWW′, respectively. This leads us to the following definition for FWB′:

$$\text{FWB}'(i) = \begin{cases} \text{empty} & i = n+1 \\ \text{FWW}'(n) & i = n \\ \text{FWB}'(i+1) \cup \{\text{FB}'(j) \mid j \in \text{FWB}'(i+1) \text{ and } j \notin \text{FWB}'(i+2)\} & (n-i) \bmod 2 = 1 \text{ and } i < n \\ \text{FWB}'(i+1) \cup \{\text{FW}'(j) \mid j \in \text{FWB}'(i+1) \text{ and } j \notin \text{FWB}'(i+2)\} & (n-i) \bmod 2 = 0 \text{ and } i < n. \end{cases}$$

Similarly, we have FBW′:

$$\text{FBW}'(i) = \begin{cases} \text{empty} & i = n+1 \\ \text{FBB}'(n) & i = n \\ \text{FBW}'(i+1) \cup \{\text{FW}'(j) \mid j \in \text{FBW}'(i+1) \text{ and } j \notin \text{FBW}'(i+2)\} & (n-i) \bmod 2 = 1 \text{ and } i < n \\ \text{FBW}'(i+1) \cup \{\text{FB}'(j) \mid j \in \text{FBW}'(i+1) \text{ and } j \notin \text{FBW}'(i+2)\} & (n-i) \bmod 2 = 0 \text{ and } i < n. \end{cases}$$

Approximations FWB′ and FBW′ are formed in the same manner as approximations FBW and FWB, respectively, with the roles of BLACK and WHITE (i.e., GB and GW) interchanged and we obtain the following upper bounds.

TABLE I. Summary of the Upper Bounds on the Number of Nodes Required for the Various Approximations

Approximation	Root type GB (GB′)	Root type GW (GW′)
FBW	$W + 1$	$W - 1$
FWB	W	W
FBW′	B	B
FWB′	$B - 1$	$B + 1$

THEOREM 4.
For the FWB′ approximation, $F \leq B + 1$ when the root is of type GW′ and $F \leq B - 1$ when the root is of type GB′.

THEOREM 5.
For the FBW′ approximation, $F \leq B$.

Theorems 4 and 5 are interesting because they bring us back a full circle to our starting point. Recall that we mentioned that a quadtree can be encoded just by listing its BLACK blocks. We say that this method did not lead to a useful approximation. Subsequently, we defined sequences of approximations based on the notion of a forest that also have the property that they never require more nodes than merely listing the BLACK nodes. Of course, Theorems 2 and 3 yield the analogous results were we to use WHITE blocks to encode the quadtree. A summary of these results is given in Table I.

The compression that is attainable as a result of using FBW and its variants reinforces our definition of a WHITE forest in terms of GB and GW in Section 2. Recall that we could have defined a WHITE forest to be analogous to a BLACK forest—that is, an internal node was of type GW′ if at least two of its sons are WHITE or of type GW′ rather than GW which required that at least three of its sons must be WHITE or of type GW. If we would have used the GW′ definition, we would constantly have to relabel the tree as we encode the tree using FBW. More importantly, our FBW approximation and its variants would not have the compression properties of Theorems 2–5. For example, the FBW encoding of a checkerboard would require us to list all of the intermediate GRAY nodes as well as all of the terminal nodes of one color.

6. EMPIRICAL RESULTS

The various encoding schemes discussed in Sections 3 and 4 were applied to a 512 × 512 image (i.e., $n = 9$) consisting of a floodplain used in prior experiments with quadtrees [25]. This floodplain is shown in Figure 11 (p. 986). A quadtree encoding of the image contains 2235 BLACK nodes and 2452 WHITE nodes. Table II (p. 986) contains a summary of the results for the FBW and FBW′ approximations. No results are tabulated for the

FWB and FWB′ approximations because their node counts will differ by one. To see this, we consider the two possible cases depending on the type of the root. When the root is of type GB, FBW(n) = {root} and FBW($n-1$) = {root} + FWB(n). When the root is of type GW, FWB(n) = {root} and FWB($n-1$) = {root} + FBW(n). An analogous statement can be made with respect to the FBW′ and FWB′ approximations. Since the approximations alternate between BLACK and WHITE nodes, our table specifies the counts for them as well as the total number of nodes.

Table II correlates with our theoretical results with respect to upper bounds on the number of nodes necessary. In particular, we find that FBW′ requires 1704 nodes to encode the image while FBW requires 1796 nodes. Thus, comparing these counts with the minimum of the BLACK and WHITE nodes in the quadtree (i.e., 2235 BLACK nodes), we find that FBW′ leads to 23.8 percent fewer nodes while FBW leads to 19.6 percent fewer nodes. These compression factors increase considerably as larger images are used (i.e., greater than $2^9 \times 2^9$ as in this example). Figures 12 and 13 (p. 988) give an example of what the images corresponding to these approximations look like. Figure 12 corresponds to FBW(n), FBW($n-1$), FBW($n-2$), and FBW($n-3$) while Figure 13 corresponds to FBW′(n), FBW′($n-1$), FBW′($n-2$), and FBW′($n-3$). The idea is to compare approximations when they contain a similar number of nodes, that is, 939 for FBW($n-1$) and 984 for FBW′(n). From Table II we also observe that FBW(n) and FBW′(n) have a different number of nodes. This is because of the different definition of GB. Recall, that for FBW, GB corresponds to at least two sons of type GB or BLACK terminal nodes; whereas for FBW′, we use GB′ which corresponds to at least three sons of type GB′ or BLACK terminal nodes. Thus, it should be clear that the GB′ criterion of FBW′ is harder to satisfy than the GB criterion of FBW thereby causing the initial approximation FBW′(n) to contain nodes from lower levels in the tree (and hence more of them!).

We also discussed approximations IB, OB, and FBB. Table III (p. 989) contains a summary of this data. Note that entries such as IB(8) = 0 and OB(8) = 4 imply 0 BLACK nodes and 4 WHITE nodes at level 8. Actually, if merges would have occurred, then we would have 0 BLACK and 1 WHITE node at level 9. However, our goal is to examine the approximation of level 8 and

TABLE II. Summary of FBW and FBW′ Approximations for Figure 11 ($n = 9$)

I	FBW(I)			FBW′(I)		
	BLACK	WHITE	TOTAL	BLACK	WHITE	TOTAL
9	39	0	39	984	0	984
8	39	900	939	984	242	1226
7	255	900	1155	1376	242	1618
6	255	1422	1677	1376	294	1670
5	325	1422	1747	1405	294	1699
4	325	1468	1793	1405	297	1702
3	328	1468	1796	1407	297	1704

FIGURE 11. Floodplain Image

these nodes are WHITE because some part of the space spanned by them was WHITE in accordance with the definition of OW. Figures 14, 15, and 16 (p. 989) correspond to IB(2), OB(2), and FBB($n-3$), respectively. Our goal is to compare FBB and OB when approximately the same number of nodes are transmitted. This is impossible for IB because it treats a GRAY node as WHITE if there is any WHITE pixel within it. In this case, we show the IB approximation at the same level as that for OB. The following observations should be apparent. First, as expected, the FBB approximation is less "blocky" at the edge than OB or IB. Second, IB does not preserve connectivity whereas OB and FBB do so at the possible expense of creating holes where there may not be any as is seen in Figure 16. These observations are not surprising in light of the above comments about IB underestimating the BLACK region.

Note that the FBW and FBW′ approximations have connectivity problems similar to IB. In particular, FBW alternates between GB and GW nodes. At iterations that use GW nodes, connectivity may be destroyed. As an example, consider Figure 17 (p. 990). Its FBW approximations are shown in Figure 18 (p. 990). We see that FBW(3) = {A}, FBW(2) = {A, 9, 13, F, G, D, E}, and FBW(1) = {A, 9, 13, F, G, D, E, 3, 5, 7, 8} = image. Figure 18b shows that FBW(2) results in the loss of connectivity. It should be clear that FWB has the same problem with respect to connectivity while iterations using GW′ cause problems for FBW′ and FWB′.

On the other hand, FWB′ approximations lead to the creation of spurious holes in the same way as does OB. In particular, FWB′ alternates between GW′ and GB′ nodes. At iterations that use GB′ nodes, spurious holes may result. As an example, consider Figure 19 (p. 991). Its FWB′ approximations are shown in Figure 20 (p. 991). We see that FWB′(3) = {A}, FWB′(2) = {A, 13, 14, 15, 21, 22, C}, and FWB′(1) = {A, 13, 14, 15,

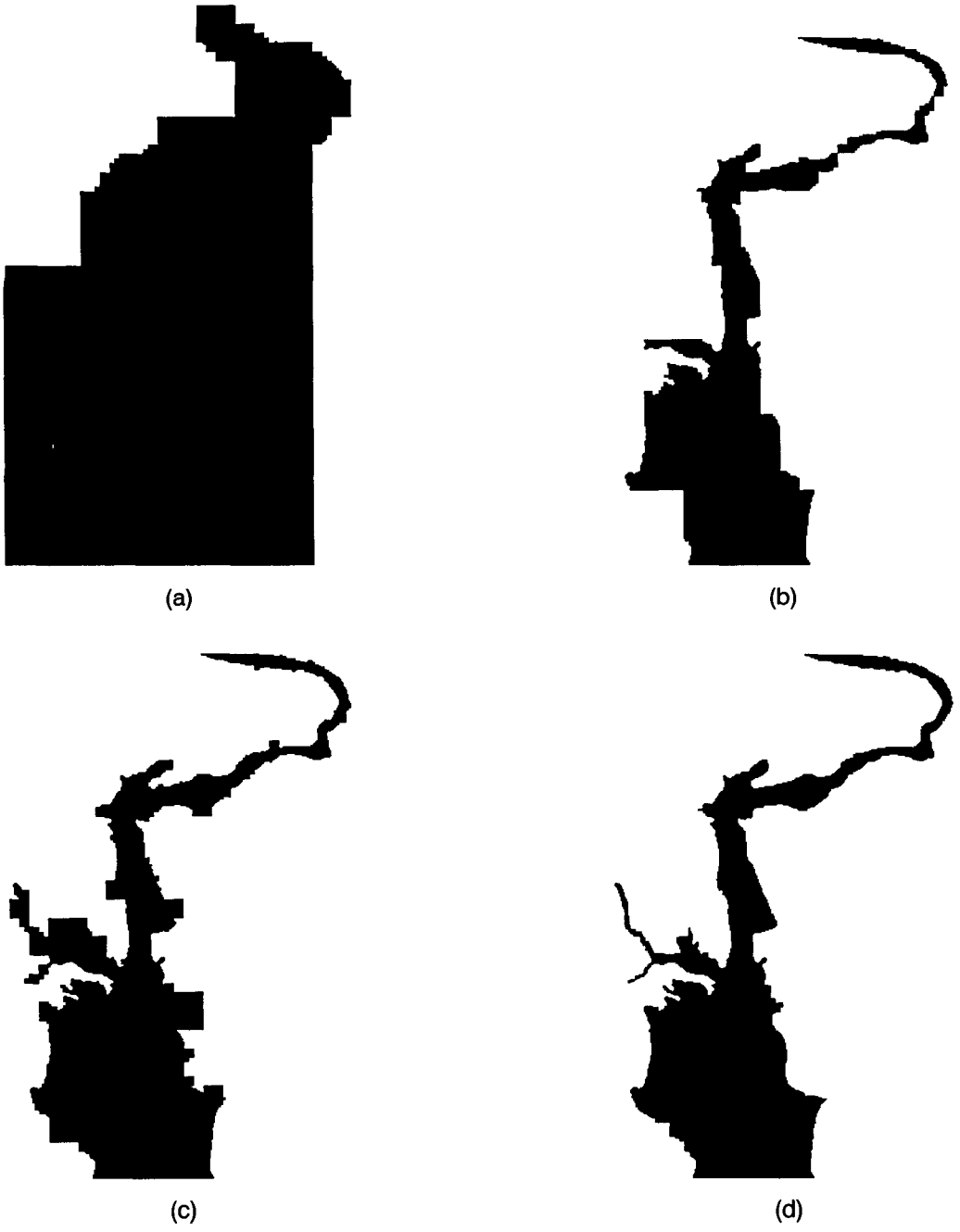


FIGURE 12. FBW Approximation. (a) FBW(n) = FBW(9). (b) FBW(n − 1) = FBW(8). (c) FBW(n − 2) = FBW(7). (d) FBW(n − 3) = FBW(6).

21, 22, C, 5, 6} = image. Figure 20b shows that FWB′(2) results in the creation of a spurious hole. It should be clear that FBW′ has the same problem with respect to the creation of a spurious hole. However, it can be shown that FBW and FWB cannot create spurious holes.

7. CONCLUDING REMARKS

A number of methods for representing quadtrees of images without space for links have been presented. In particular, approximation techniques were demonstrated which were superior to merely listing the BLACK nodes (or just the WHITE nodes). Furthermore, these approximations were also seen to lead to compression in the sense that the number of nodes required was always less than or equal to MIN(B, W). There are a number of reasons for the success of the FBW approximation (we shall use the term FBW to mean FBW and its variants FWB, FBW′, and FWB′). First, FBW yields a saving of space whenever the situation arises that 3 out of 4 sons have the same type (i.e., BLACK or WHITE). This, coupled with the alternation between BLACK and

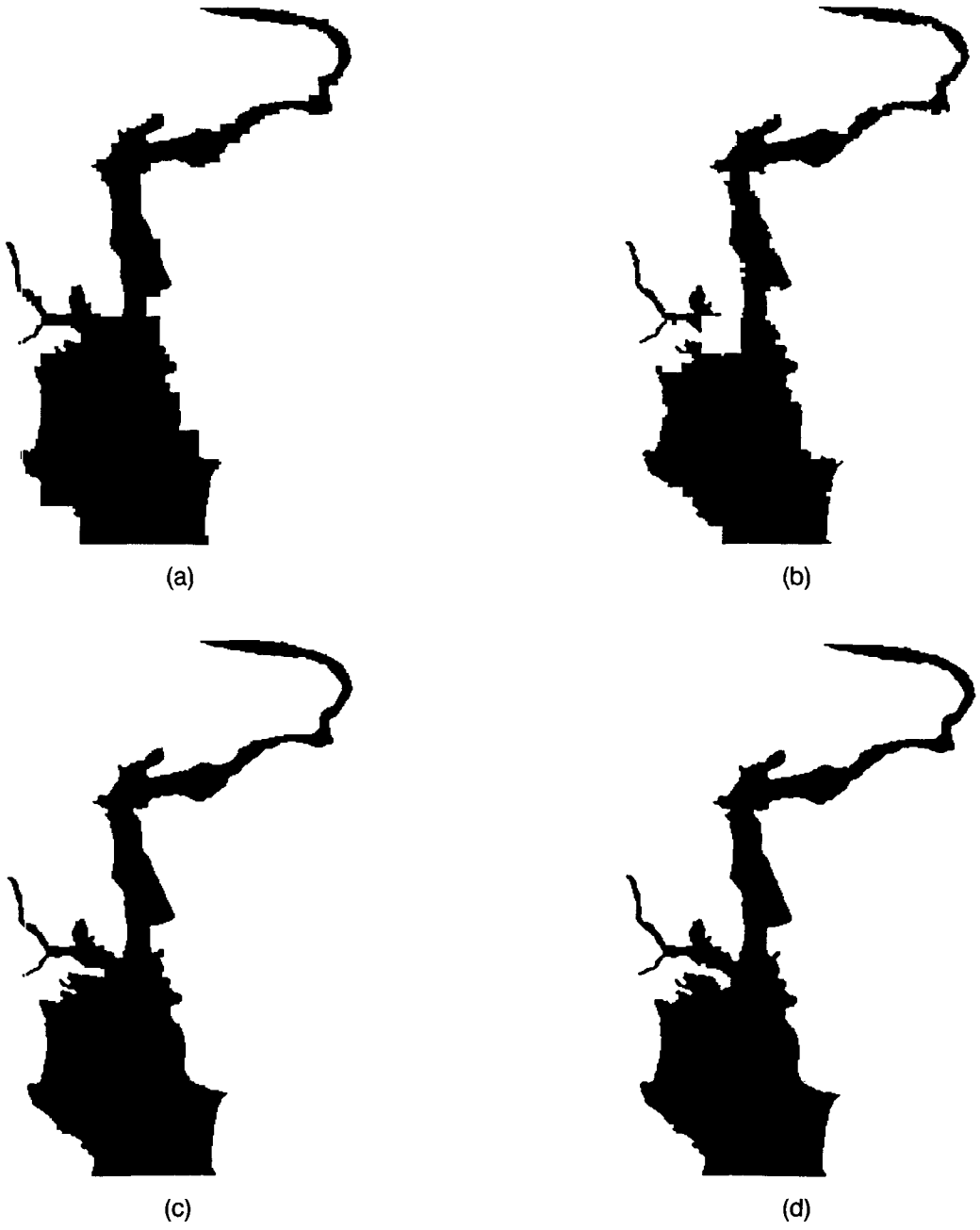


FIGURE 13. FBW′ Approximation. (a) FBW′(*n*) = FBW′(9). (b) FBW′(*n* − 1) = FBW′(8). (c) FBW′(*n* − 2) = FBW′(7). (d) FBW′(*n* − 3) = FBW′(6).

WHITE forests, enables the approximation to zoom in on the final goal. Second, the encoding and decoding procedures ENCODE_FBW and BUILD_TREE enable us to have an encoding which makes use of BLACK, WHITE, and GRAY nodes without needing to specify their type. In addition, these procedures are very efficient. ENCODE_FBW takes time proportional to the size of the quadtree being encoded since it is nothing but a tree traversal. BUILD_TREE takes time proportional to the product of the number of nodes in the FBW encoding and the resolution of the image (i.e., n for a $2^n \times 2^n$ image) since the node is represented by an n digit base 5 code indicating the path from the root which must be decoded.

The FBW approximation was shown to yield compression. Clearly, the compression increases with the frequency of the occurrence of 3 out of 4 sons of the same color at different levels of the tree (e.g., Figure 10c). It is desirable to categorize the class of images where the compression factor is a maximum. Other interesting questions include the determination of the average amount of compression for a $2^n \times 2^n$ image as a

function of n. Similarly, what is the maximum compression factor for an image as a function of n? Note that the Quadtree Medial Axis Transform [23] also exhibits compression albeit in a different way.

Use of FBW as an approximation method is superior to the inner and outer approximations of [18]. The approximation is biased in favor of objects with so called "panhandles" rather than "staircases" as shown in Figures 21a and 21b (p. 991). The effectiveness of the approximation could be evaluated by defining a measure such as "nodes"/"area in the approximation" or, even better, some measure that reflects the amount of area by which members of the sequence of approximations underestimate or overestimate the true area.

TABLE III. Summary of IB, OB, and FBB Approximations for Figure 11 ($n = 9$)

i	IB(i)/GW(i)	OB(i)/IW(i)	FBB(i)
9	0/1	1/0	39
8	0/4	3/1	109
7	0/13	9/4	265
6	1/39	23/17	537
5	8/98	63/43	878
4	27/224	149/122	1308
3	118/372	372/265	1745
2	367/1032	846/553	2079
1	1008/1828	1625/1211	2235
0	2235/2452	2235/2452	

FIGURE 14. IB(2)

It would be desirable to compare FBW with other representations such as run lengths, boundary codes, arrays, quadtrees, etc. How hard is it to perform basic image processing operations given an FBW encoding? If we know the type of each node (i.e., BLACK, WHITE, or GRAY), then we can compute neighbors, in the sense of [22], in the various directions with some work. The concept of a locational code is useful in representing a map consisting of polygons (e.g., a county map) where the map is stored as a collection of quadtrees. There is one quadtree per polygon and what is stored is a locational code to the root of the smallest enclosing square for the polygon. This is currently being used in a cartographic database project [25]. Labeling internal nodes of a quadtree with GB and GW to indicate information about the subtrees may find application in using quadtrees for matching of images. There is no need for extra storage since a type field already exists for each node. Nonterminal nodes can be distinguished from terminal nodes by the fact that the latter have four empty sons. Thus, internal nodes can also be labeled BLACK or WHITE corresponding to GB and GW, respectively.

FIGURE 15. OB(2)

FIGURE 16. FBB($n - 3$) = FBB(6).

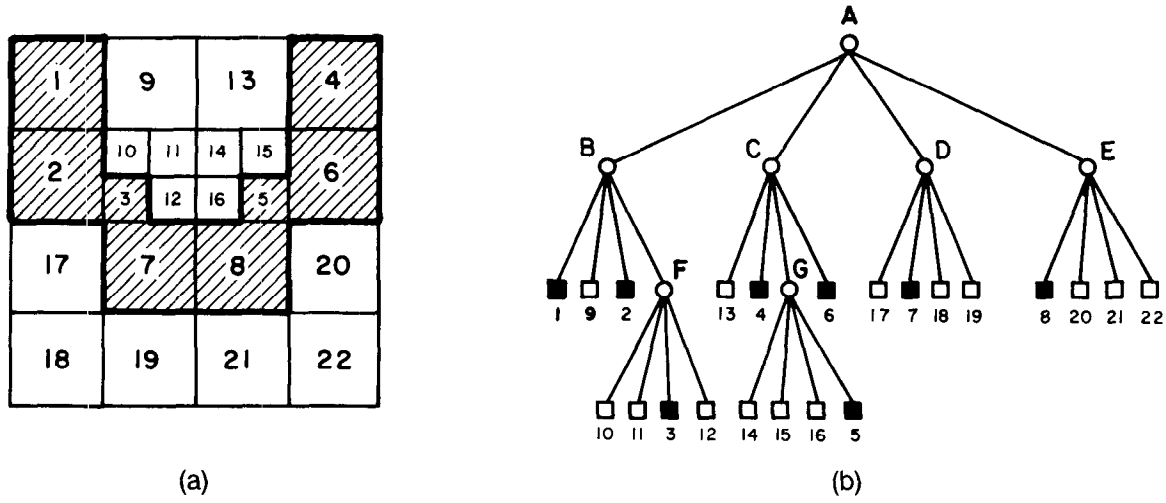


FIGURE 17. (a) Sample Image and (b) Its Quadtree that Demonstrates Loss of Connectivity by using the FBW Approximation

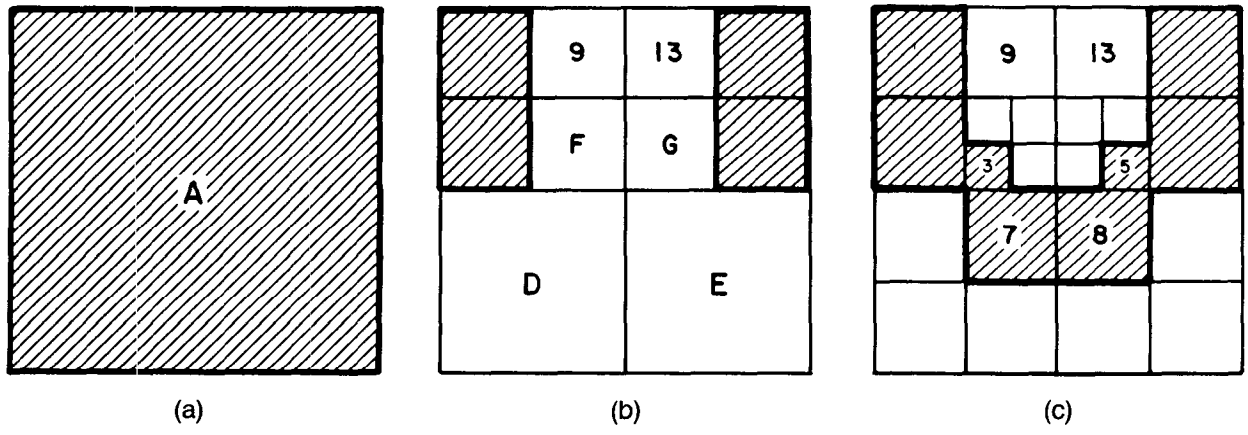


FIGURE 18. (a) FBW(3), (b) FBW(2), and (c) FBW(1) for Figure 17

APPENDIX

THEOREM 2.

The maximum number of nodes in an FBW approximation is less than or equal to one plus the number of WHITE nodes in the quadtree (i.e., $F \leq W + 1$).

PROOF.

Our proof makes use of induction on the tree size and also on the sequence of approximations comprising FBW. It is clear that the theorem holds for the empty tree as well as all variants of a root and four sons which are terminal nodes (e.g., Figures 22a–22e (p. 992)). This forms the base case. Our proof consists of two cases depending on whether the root is of type GB or type GW. Assume a $2^n \times 2^n$ image.

Case a: The root is of type GB. Therefore, FBW(n) = {root}. For this discussion, we shall use type GW to include terminal WHITE nodes as well and likewise for GB and terminal BLACK nodes. Define FBWN(i) = $\{j \mid j \in \text{FBW}(i)$ **and** $j \notin \text{FBW}(i+1)\}$ for $0 \leq i < n$. For each element of FBW(n), say p, FBWN($n-1$) contains all descendants of p of type GW that do not have an ancestor of type GW in the tree rooted at p (i.e., nodes 1, 15, 33, and 34 in Figure 1). Suppose FBWN($n-1$) has m elements. For each element of FBWN($n-1$) that is not a terminal node, say x_i, FBWN($n-2$) contains all descendants of x_i of type GB that do not have an ancestor of type GB in the tree rooted at x_i (i.e., node 19 in Figure 1). Let the tree rooted at node x_i contribute y_i nodes of type GB to FBWN($n-2$). The tree rooted at node x_i has at least $3 \cdot y_i + 1$ nodes of type GW. To see this, we observe that each GB node contributed by x_i must have three GW brothers since its father is of type GW. The 1 is contributed by node x_i since it is itself of type GW. Of these $3 \cdot y_i + 1$ nodes, y_i of them must be nonterminal nodes of type GW where each such node corresponds to the father of one of the y_i GB nodes. Of course, these fathers are unique—that is, a node can only have one father. Therefore, $3 \cdot y_i + 1 - y_i = 2 \cdot y_i + 1$ of the GW nodes are terminal WHITE nodes which have been subsumed by our construction of FBWN($n-2$). Two items are worthy of note. First, by

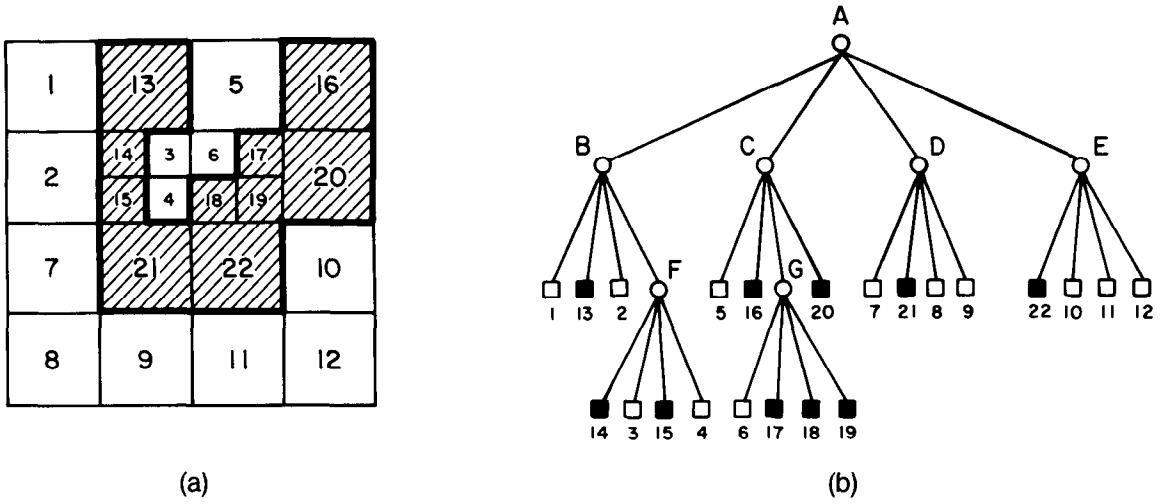


FIGURE 19. (a) Sample Image and (b) Its Quadtree that Demonstrates Spurious Holes by Using the FBW′ Approximation

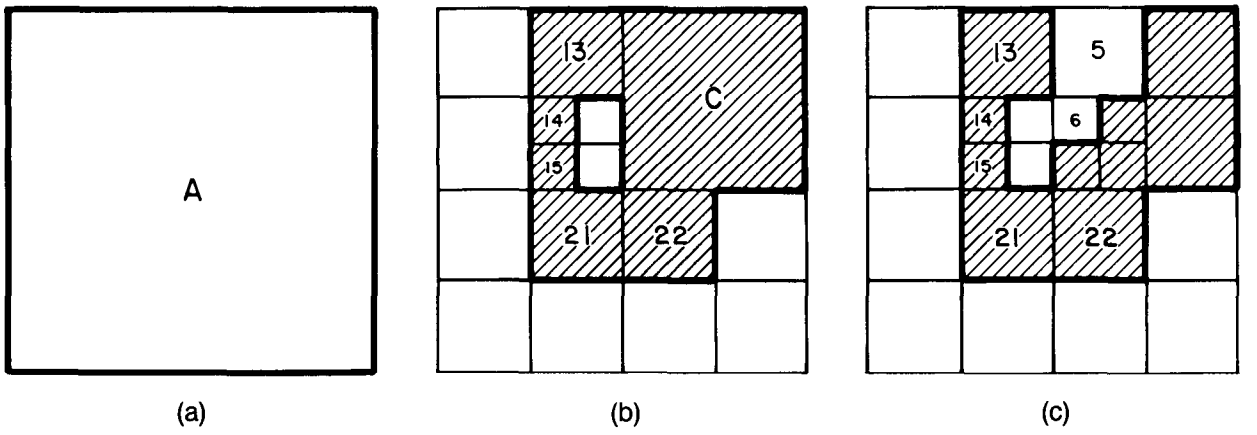


FIGURE 20. (a) FBW′(3), (b) FBW′(2), and (c) FBW′(1) for Figure 19

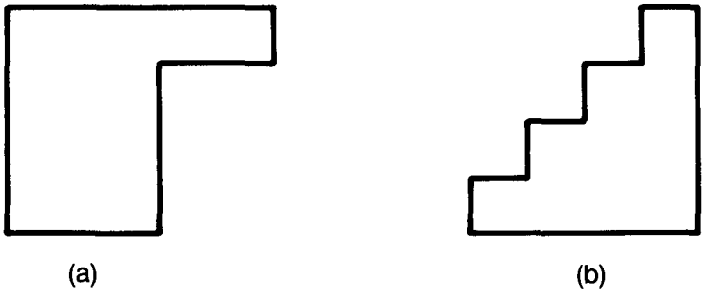


FIGURE 21. Sample Image Illustrating the Biases of the FBW Approximation: (a) Panhandle, (b) Staircase

use of the term *subsumed* we are also including nodes already in FBWN($n-1$) when $y_i = 0$. Second, $2 \cdot y_i + 1$ is a lower bound on the number of terminal WHITE nodes which are subsumed. For example, in Figure 23 (p. 992), FBW(n) = {A}, FBWN($n-1$) = {B}, FBWN($n-2$) = {4, 5, 9, 16}. The tree rooted at B contributes $y_i = 4$ nodes of type GB to FBWN($n-2$) and it has 17 nodes of type GW of which $12 > 2 \cdot y_i + 1$ are terminal WHITE nodes. The next step is to construct the sequence FBWN($n-3$) through FBWN(0). Let $w(z)$ denote the number of WHITE nodes in the tree rooted at node z. We use induction to note that for each element of FBWN($n-2$), say z, its FBW approximation has $\leq w(z) + 1$ nodes. Therefore, the FBW approximation for

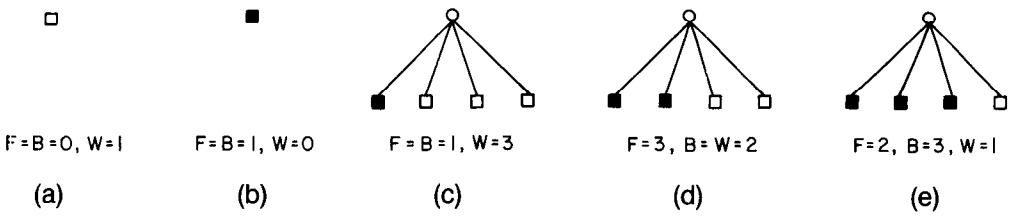


FIGURE 22. Variants of All Quadtrees for a $2^1 \times 2^1$ Image

our original tree consists of at most Q elements where Q is

$$Q = 1 + m + \sum_{i=1}^{m} y_i + \sum_{z \in \mathrm{FBWN}(n-2)} (w(z) + 1)$$

Letting $P = \sum_{i=1}^{m} y_i$ we can rewrite Q as

$$Q = 1 + m + P + \sum_{z \in \mathrm{FBWN}(n-2)} w(z) + P$$

$$= 1 + 2 \cdot P + m + \sum_{z \in \mathrm{FBWN}(n-2)} w(z).$$

But $2 \cdot P + m$ is a lower bound on the number of WHITE terminal nodes in the tree subsumed by the construction of FBWN($n - 2$) which means that $Q \leq W + 1$ and our theorem holds when the root of the tree is of type GB. Note that in our proof it makes no difference whether or not elements of FBWN($n - 1$) are terminal WHITE nodes.

Case b: The root is of type GW. This is analogous to treating a fictitious FBW($n + 1$) = {root} and constructing FBW(n) in the manner done in Case a. Thus, since there is only one GW node in FBW($n + 1$) (i.e., root), we say that the root contributes y nodes of type GB that do not have an ancestor of type GB to FBW(n). Using the same reasoning as in Case a we have at least $2 \cdot y + 1$ terminal WHITE nodes which are subsumed by the construction of FBW(n). Next, construct the sequence FBW($n - 1$) through FBW(0). Let $w(z)$ denote the number of WHITE nodes in the tree rooted at node z. We use induction by appealing to Case a to note that for each element of FBW(n), say z, its approximation has $\leq w(z) + 1$ nodes. Therefore, the FBW approximation for our original tree consists of at most Q elements where Q is

$$Q = y + \sum_{z \in \mathrm{FBW}(n)} (w(z) + 1)$$

$$= y + \sum_{z \in \mathrm{FBW}(n)} w(z) + y$$

$$= 2 \cdot y + \sum_{z \in \mathrm{FBW}(n)} w(z).$$

But $2 \cdot y + 1$ is a lower bound on the number of WHITE terminal nodes in the tree subsumed by the construction of FBW(n) which means that $Q = W - 1$ and our theorem holds when the root of the tree is of type GW.

THEOREM 3.
The maximum number of nodes in an FWB approximation is less than or equal to the number of WHITE nodes in the quadtree (i.e., $F \leq W$).

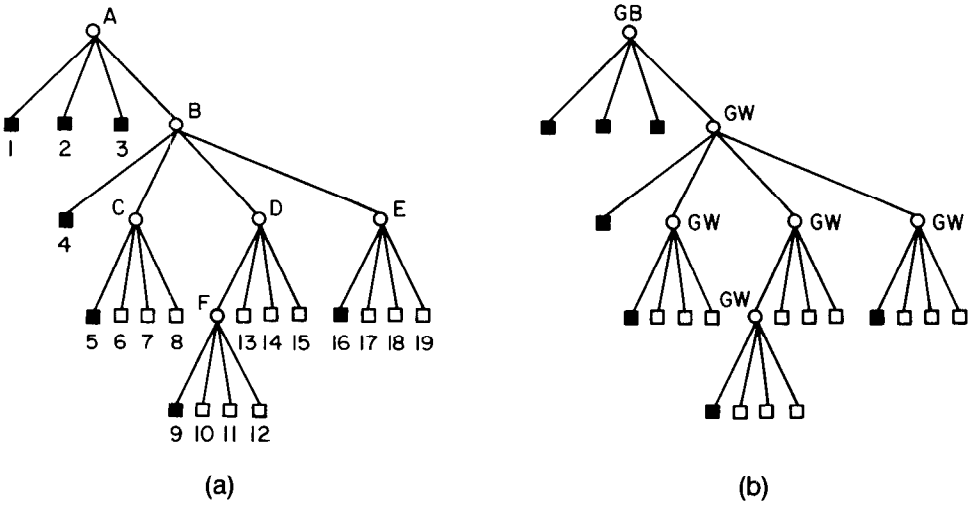


FIGURE 23. (a) Sample Quadtree and (b) Its GB/GW Encoding Illustrating the Amount of Subsumption of Nodes Attainable by Using FBW

PROOF.
Our proof relies heavily on the proof for the FBW approximation of Theorem 2. It consists of two cases depending on whether the root is of type GB or GW. Assume a $2^n \times 2^n$ image.

Case a: The root is of type GB. Therefore, FWB(n) ignores the root and now proceeds to collect all descendants of the root of type GW that do not have an ancestor of type GW (as in Theorem 2, we use the type GW to include terminal WHITE nodes as well and likewise for GB and terminal BLACK nodes). But as we saw in the proof of Case a of Theorem 2, these are precisely the nodes that comprise FBW($n - 1$). In other words, {root} + FWB(n) = FBW($n - 1$). The rest of the sequence is constructed in an analogous manner—that is, {root} + FWB(i) = FBW($i - 1$) for $1 \leq i \leq n$. Approximation FBW has an upper bound of $W + 1$ nodes and since FWB does not include the root node we find that the size of FWB is bounded from above by W.

Case b: The root is of type GW. Therefore, FWB(n) includes the root and now proceeds to form FWB($n - 1$). FWB($n - 1$) consists of the root and all descendants of the root of type GB that do not have an ancestor of type GB. But as we saw in the proof of Case b of Theorem 2, these are precisely the nodes that comprise FBW(n). In other words, FWB($n - 1$) = {root} + FBW(n). The rest of the sequence is constructed in an analogous manner—that is, FWB($i - 1$) = {root} + FBW(i) for $1 \leq i \leq n$. Approximation FBW has an upper bound of $W - 1$ nodes and since FWB also includes the root node we find that the size of FWB is bounded from above by W.

ACKNOWLEDGMENTS

I would like to thank Robert E. Webber for his valuable comments and criticisms, and Yuangeng Huang for generating Figures 11–16 and Tables II and III.

REFERENCES

1. Abel, D.J., and Smith, J.L. A data structure and algorithm based on a linear key for a rectangle retrieval problem. *Comput. Vision, Graphics, and Image Process. 24*, 1 (Oct. 1983), 1–13.
2. Burton, F.W., and Kollias, J.G. Comment on the explicit quadtree as a structure for computer graphics. *Comput. J. 26*, 2 (May 1983), 188.
3. Cook, B.G. The structural and algorithmic basis of a geographic data base. In *Proceedings of the First International Advanced Study Symposium on Topological Data Structures for Geographic Information Systems*. G. Dutton, Ed., Harvard Papers on Geographic Information Systems, 1978.
4. Dyer, C.R., Rosenfeld, A., and Samet, H. Region representation: Boundary codes from quadtrees. *Commun. ACM 23*, 3 (Mar. 1980), 171–179.
5. Gargantini, I. An effective way to represent quadtrees. *Commun. ACM 25*, 12 (Dec. 1982), 905–910.
6. Hunter, G.M. Efficient computation and data structures for graphics. Ph.D. dissertation, Dept. of Electrical Engineering and Computer Science, Princeton Univ., N.J., 1978.
7. Hunter, G.M., and Steiglitz, K. Operations on images using quad trees. *IEEE Trans. Pattern Anal. Machine Intell. 1*, 2 (Apr. 1979), 145–153.
8. Hunter, G.M., and Steiglitz, K. Linear transformation of pictures represented by quad trees. *Comput. Graphics and Image Process. 10*, 3 (July 1979), 289–296.
9. Jackins, C.L., and Tanimoto, S.L. Oct-trees and their use in representing three-dimensional objects. *Comput. Graphics and Image Process. 14*, 3 (Nov. 1980), 249–270.
10. Jones, L., and Iyengar, S.S. Representation of regions as a forest of quadtrees. In *Proceedings of the IEEE Conference on Pattern Recognition and Image Processing*, Dallas, Tex., 1981, pp. 57–59.
11. Kawaguchi, E., and Endo, T. On a method of binary picture representation and its application to data compression. *IEEE Trans. Pattern Anal. Machine Intell. 2*, 1 (Jan. 1980), 27–35.
12. Klinger, A. Patterns and search statistics. In *Optimizing Methods in Statistics*. J.S. Rustagi, Ed., Academic Press, NY, 1971, pp. 303–337.
13. Klinger, A., and Rhodes, M.L. Organization and access of image data by areas. *IEEE Trans. Pattern Anal. Machine Intell. 1*, 1 (Jan. 1979), 50–60.
14. Knowlton, K. Progressive transmission of grey-scale and binary pictures by simple, efficient, and lossless encoding schemes. In *Proceedings of the IEEE 68*, 7 (July 1980), pp. 885–896.
15. Meagher, D. Geometric modeling using octree encoding. *Comput. Graphics and Image Process. 19*, 2 (June 1982), 129–147.
16. Morton, G.M. A computer oriented geodetic data base and a new technique in file sequencing, IBM Canada, 1966.
17. Oliver, M.A., and Wiseman, N.E. Operations on quadtree-encoded images, *Comput. J. 26*, 1 (Feb. 1983), 83–91.
18. Ranade, S., Rosenfeld, A., and Samet, H. Shape approximation using quadtrees. *Pattern Recognition 15*, 1 (1982), 31–40.
19. Samet, H. Region representation: Quadtrees from boundary codes. *Commun. ACM 23*, 3 (Mar. 1980), 163–170.
20. Samet, H. An algorithm for converting rasters to quadtrees. *IEEE Trans. Pattern Anal. Machine Intell. 3*, 1 (Jan. 1981), 93–95.
21. Samet, H. Connected component labeling using quadtrees. *J. ACM 28*, 3 (July 1981), 487–501.
22. Samet, H. Neighbor finding techniques for images represented by quadtrees. *Comput. Graphics and Image Process. 18*, 1 (Jan. 1982), 37–57.
23. Samet, H. A quadtree medial axis transform. *Commun. ACM 26*, 9 (Sept. 1983), 680–693.
24. Samet, H. The quadtree and related hierarchical data structures. *ACM Comput. Surv. 16*, 2 (June 1984), 187–260.
25. Samet, H., Rosenfeld, A., Shaffer, C., and Webber, R.E. Quadtree region representation in cartography: Experimental results. *IEEE Trans. Syst., Man, Cybern. 13*, 6 (Nov./Dec. 1983), 1148–1154.
26. Sloan, K.R., Jr., and Tanimoto, S.L. Progressive refinement of raster images. *IEEE Trans. Comput. 28*, 11 (Nov. 1979), 871–874.
27. Tanimoto, S., and Pavlidis, T. A hierarchical data structure for picture processing. *Comput. Graphics and Image Process. 4*, 2 (June 1975), 104–119.
28. Weber, W. Three types of map data structures, their ANDs and NOTs, and a possible OR. In *Proceedings of the First International Advanced Study Symposium on Topological Data Structures for Geographic Information Systems*, G. Dutton, Ed., Harvard Papers on Geographic Information Systems, 1978.
29. Woodwark, J.R. The explicit quadtree as a structure for computer graphics. *Comput. J. 25*, 2 (May 1982), 235–238.
30. Yau, M., and Srihari, S.N. A hierarchical data structure for multidimensional digital images. *Commun. ACM 26*, 7 (July 1983), 504–515.

CR Categories and Subject Descriptors: E.1 [**Data**]: Data Structures—*trees*; E.4 [**Data**]: Coding and Information Theory—*data compaction and compression*; I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding—*representations, data structures and transforms*; I.4.2 [**Image Processing**]: Compression (Coding)—*approximate methods, exact coding*
General Terms: Algorithms, Theory
Additional Key Words and Phrases: quadtrees, hierarchical data structures, image databases, image transmission, progressive approximation.

Received 12/83; revised 9/84; accepted 3/85

Author's Present Address: Hanan Samet, Computer Science Department, University of Maryland, College Park, MD 20742.