Distance Transform for Images Represented by Quadtrees

HANAN SAMET

Abstract—The concept of distance used in binary array representations of images is adapted to a quadtree representation. The chessboard distance metric is shown to be particularly suitable for the quadtree. A chessboard distance transform for a quadtree is defined as the minimum distance in the plane from each BLACK node to the border of a WHITE node. An algorithm is presented which computes this transform by only examining the BLACK node's adjacent and abutting neighbors and their progeny. However, unlike prior work with quadtrees, computation of the distance transform requires a capability of finding neighbors in the diagonal direction rather than merely in the horizontal and vertical directions. The algorithm's average execution time is proportional to the number of leaf nodes in the quadtree.

Index Terms—Distance transforms, image processing, pattern recognition, quadtrees.

I. Introduction

The quadtree [2]-[8], [10]-[14] is a compact hierarchical representation which is spacewise efficient and also serves to facilitate search. In this correspondence we investigate the concept of distance [1], [9] and formulate a definition and metric which are applicable to quadtrees. The notion of distance is important in image processing applications for computing properties of an image such as its skeleton as well as for applying operations such as propagation and shrinking [9]. We show that the chessboard distance metric is particularly suitable for the quadtree. This leads us to develop an entity which we term the chessboard distance transform of a quadtree. An algorithm for its computation is presented.

Our algorithm requires a capability of locating adjacent nodes. One of the key differences between our work on quadtrees [2], [10]-[12] and earlier work of others [3]-[5], [8] is in the way in which we locate adjacent nodes. We do not use position (i.e., coordinates) or size information [8]. We also do not assume the existence of links (and the amount of extra space that they require) from a node to its adjacent nodes [3]-[5]. Instead, our techniques only make use of the existing structure of the tree. However, unlike our prior work, we now must find adjacent nodes in the diagonal direction as well rather than merely in the horizontal and vertical directions. We demonstrate how such diagonally adjacent nodes are found and discuss its relationship to the method of determining adjacent nodes in the horizontal and vertical directions.

II. Definitions and Notation

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

Each node in a quadtree is stored as a record containing seven fields. The first five fields contain pointers to the node's father and its four sons, labeled *NW*, *NE*, *SE*, and *SW*. Given a node P and a son I, these fields are referenced as FATHER(P) and SON(P, I), respectively. At times it is useful to use the function SONTYPE(P) where SONTYPE(P) = Q iff SON(FATHER(P), Q) = P. The sixth field NODETYPE describes the contents of the block of the image which the node represents, i.e., BLACK, WHITE, or GRAY. The seventh field DIST indicates the distance to the nearest WHITE node according to the specified distance metric. This field is only meaningful for BLACK nodes. A WHITE node is said to have distance zero.

Let the four sides of a node's block be called its *N*, *E*, *S*, and *W* sides. They are also termed its boundaries and at times we speak of them as if they are directions. Fig. 2 shows the relationship between the quadrants of a node's block and its boundaries. The specification of the spatial relationships between the various sides is facilitated by use of the functions OPSIDE, CSIDE, CCSIDE, and QUAD. OPSIDE(B) is a side facing side B, e.g., OPSIDE(N) = S. CSIDE(B) and CCSIDE(B) correspond to the sides adjacent to side B in the clockwise and counterclockwise directions, respectively, e.g., CSIDE(N) = E and CCSIDE(N) = W. QUAD($S1$, $S2$) is the quadrant bounded by boundaries $S1$ and $S2$ (if $S1$ and $S2$ are not adjacent boundaries, then QUAD($S1$, $S2$) is undefined), e.g., QUAD(S, E) = SE, while QUAD(S, N) is undefined. Similarly, OPQUAD (QUAD($S1$, $S2$) = QUAD(OPSIDE($S1$), OPSIDE($S2$)).

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

III. Distance

For an image represented by a binary array, we can define a function d that takes pairs of points into nonnegative numbers. It is called a metric or a distance function if for all points p, q, and r the following relations are satisfied:

1) $d(p, q) \geq 0$, and $d(p, q) = \phi$ if and only if $p = q$ (positive definiteness),

2) $d(q, p) = d(p, q)$ (symmetry),

3) $d(p, r) \leq d(p, q) + d(q, r)$ (triangle inequality).

Given the points $p = (p_x, p_y)$ and $q = (q_x, q_y)$ we now examine some of the more common metrics. The most commonly used metric is the Euclidean distance

$$d_E(p, q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}.$$

Manuscript received July 18, 1979; revised July 27, 1981. This work was supported by the Defense Advanced Research Projects Agency and the U.S. Army Night Vision Laboratory under Contract DAAG-53-76C-0138 (DARPA Order 3206).

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

0162-8828/82/0500-0298$00.75

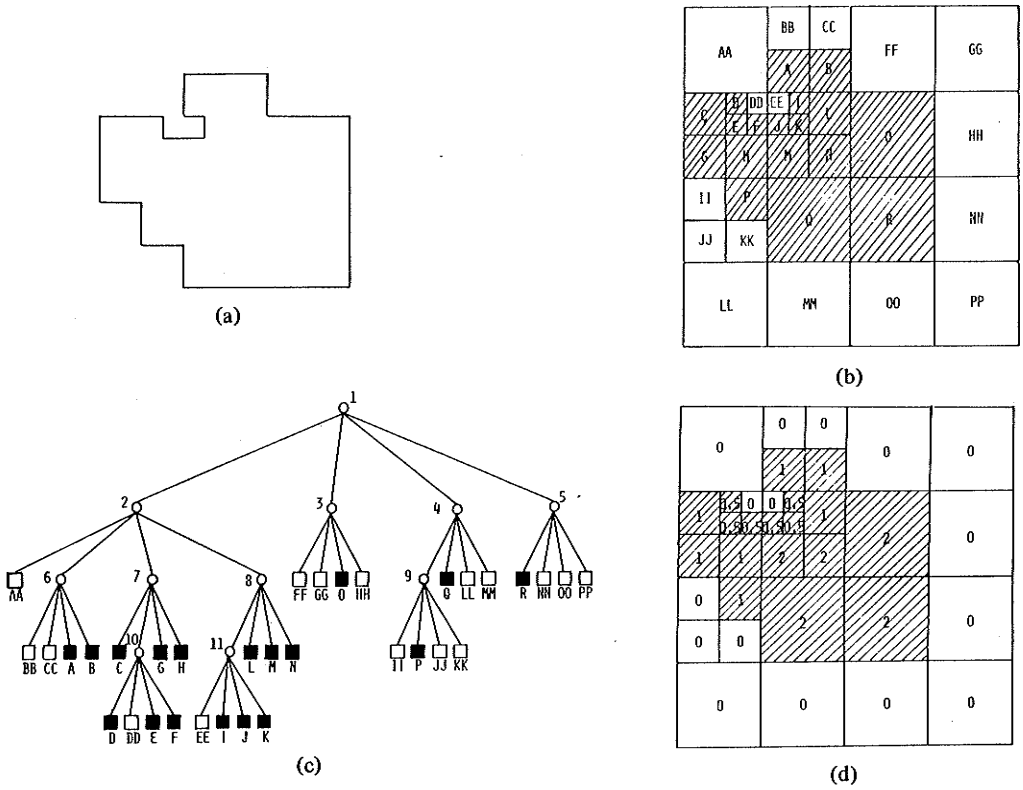


Fig. 1. An image, its maximal blocks, the corresponding quadtree, and the chessboard distance transform. Blocks in the image are shaded. (a) Sample image. (b) Block decomposition of the image in (a). (c) Quadtree representation of the blocks in (b). (d) Chessboard distance transform of (b).

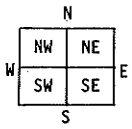


Fig. 2. Relationship between a block's four quadrants and its boundaries.

Two other metrics which are used in image processing are the absolute value metric, or the city block distance

$$d_A(p, q) = |p_x - q_x| + |p_y - q_y|$$

and the maximum value metric, or the chessboard distance

$$d_M(p, q) = \max \{|p_x - q_x|, |p_y - q_y|\}.$$

The set of points q having $d_E(p, q) \leqslant t$ are those points contained in a circle centered at p having radius t. Similarly, $d_A(p, q) \leqslant t$ yields a diamond, centered at p, with side length $t\sqrt{2}$, and $d_M(p, q) \leqslant t$ yields a square, centered at p, with side length $2t$.

For an image represented by a quadtree we use the same metrics. The only difference is that the points for which the metrics are defined are centers of blocks. We also define the distance transform T for a quadtree to be a function that yields for each BLACK block in the quadtree the distance (in the chosen metric) from the center of the block to the nearest point which is on a BLACK-WHITE border. More formally, letting x be the center of a BLACK block B, z be a point on the border of a WHITE block W, and only using F intermediately as the distance to a particular WHITE block's border, we have

$$F(B, W) = \min_z d(x, z)$$

$$T(B) = \min_W F(B, W).$$

We say that T of a WHITE block is zero and that the border of the space represented by the quadtree of the image is BLACK. Notice that the distance transform is not defined in terms of a center to center distance. This is done to avoid a bias against large size WHITE adjacent blocks, and moreover it will be seen to enable us to restrict the number of nodes visited while computing it.

Given blocks P and Q, we say that Q is a *neighbor* of P when both of the following conditions are satisfied:

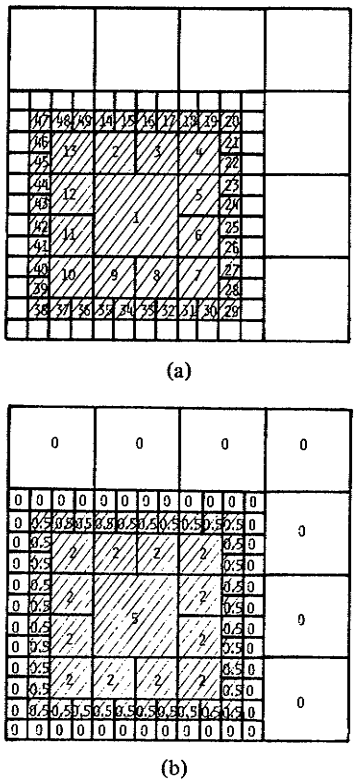
1) P and Q share a common border, even if only a corner,

2) if Q is a BLACK or WHITE block, then its size is greater than or equal to that of P, while if it is a GRAY block, then it and P are of equal size.

For example, block R in Fig. 1(b) and (c) has neighbors O, NN, OO, Q, HH, PP, MM, and 8. A block has a maximum of eight neighbors, in which case they are all of equal size [e.g., block O in Fig. 1(b)], and a minimum of five neighbors. The minimum is obtained by observing that a node cannot be adjacent to two nodes of greater size on opposite sides (e.g., given node P, and nodes Q and R adjacent to its east and west sides, respectively, then nodes Q and R cannot be both greater in size than P). Thus, a node can have at most two larger sized neighbors adjacent to its nonopposite sides and two of these neighbors can subsume at most three additional neighbors, thereby requiring at least three more neighbors. For example, for node D in Fig. 1(b), AA subsumes the NW, N, and NE neighbors; C subsumes the W and SW neighbors; the remaining neighbors are E, F, and DD in directions S, SE, and E, respectively. We can now prove the following theorem which aids in understanding the amount of work involved in computing the distance transform for any metric.

Theorem 1: For any BLACK block in the image, its neighbors cannot all be BLACK.

Proof: One of the neighbors of the block, say P, must be GRAY or WHITE since otherwise merging would have taken



(a)

(b)

Fig. 3. An image illustrating the maximum number of nodes that need to be visited when computing the chessboard distance transform value for node 1. (a) Image. (b) Chessboard distance transform of the image in (a).

place and P would not be in the image (i.e., P would be part of a bigger BLACK block). Recall that neighbors are adjacent or abutting blocks of greater than or equal size. Q.E.D.

Theorem 1 makes the chessboard distance metric especially attractive. It means that for a BLACK block of size 2^S, say P, the center of the WHITE block whose border is nearest to P (hereinafter referred to as the nearest WHITE block) must be found at a distance of $< 3 \cdot 2^{S-1}$, i.e., within a square centered at P of side length $3 \cdot 2^S$. In fact, the worst case arises when the nearest WHITE block is a block of minimum size (i.e., a single pixel) adjacent to the furthest boundary of P's neighboring GRAY block [e.g., WHITE block EE with respect to BLACK block N in Fig. 1(b)]. Notice that none of P's neighboring BLACK blocks need be taken into consideration in computing P's chessboard distance transform since the value would have to be at least $3 \cdot 2^{S-1}$ (e.g., BLACK blocks O, Q, and R with respect to BLACK block N in Fig. 1(b). Thus, Theorem 1 means that when computing the chessboard distance transform of a quadtree, for each node corresponding to a BLACK block we only need to consider its GRAY neighboring nodes. Fig. 3(a) illustrates the worst case in terms of the number of blocks that need to be examined, i.e., BLACK block 1 is surrounded by rings of BLACK blocks of decreasing size.

Theorem 1 can also be used to constrain the amount of work needed to compute other distance transforms. In the case of the Euclidean distance transform given BLACK node P of size 2^S, the nearest WHITE block is at a distance $< 3 \cdot 2^{S-1} \cdot \sqrt{2}$. Similarly, when a city block distance transform is used the maximum distance is $< 3 \cdot 2^S$. These values are all derived analogously.

Unfortunately, we cannot say that larger sized neighbors need not be taken into consideration when computing the Euclidean and city block distance transforms. That this is true can be seen by examining Fig. 4 which illustrates the regions

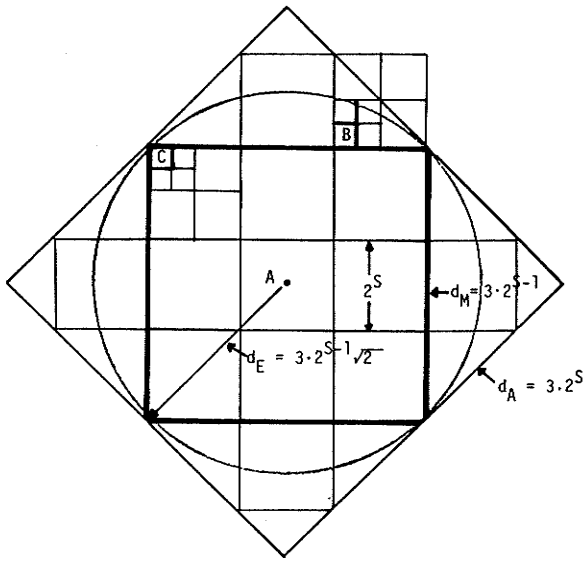


Fig. 4. Regions within which the closest WHITE node to BLACK node A must lie for several metrics.

within which Theorem 1 stipulates that the nearest WHITE block be found. For example, given BLACK block A of size 2^S, in the case of both the Euclidean and city block distance, block B may be the nearest WHITE block to block A. This may require visiting an eastern neighbor of block A of size 2^{S+1}. On the other hand, when chessboard distance is used, block C is the nearest to block A and, in fact, no neighboring blocks of greater size ever need to be visited. Thus, we see that the Euclidean and city block distances may lead to more than eight neighboring blocks of equal size being visited or even to blocks of greater size.

Note that our definition of distance treats nonexistent neighbors (i.e., on the other side of the border of the space represented by the quadtree) as BLACK and of equal size. This is consistent with the definition of the chessboard distance transform as yielding for each BLACK node in the quadtree the distance from the center of the block to the nearest point which is on the border between a BLACK and a WHITE node.

In the remainder of this correspondence we focus our attention on the chessboard distance transform due to its computational simplicity. This simplicity arises from the property of the chessboard distance metric that the set of points q such that $d(p, q) \leqslant t$ is a square, rather than a circle or a diamond as is true for the Euclidean and city block distance metrics, respectively.

IV. Locating Neighbors

The chessboard distance transform computation algorithm requires us to be able to examine the eight neighbors of each BLACK node. Thus, we must be able to locate these neighbors. This is different from our own earlier work on quadtrees [2], [10]-[12] since there we were only concerned with determining nodes in the horizontal and vertical direction (i.e., N, E, S, and W), whereas now we must also be able to locate neighboring nodes in the diagonal direction (i.e., NE, SE, SW, and NW). However, unlike earlier work of others on quadtrees, we do not use position (i.e., coordinates) or size information [8]. We also do not assume the existence of links (and the amount of extra space that they require) from a node to its adjacent nodes [3]-[5]. Instead, we only make use of the existing structure of the tree. In the following, we first review how neighbors in the horizontal and vertical direction are found. Next, we show how these methods can be extended to enable the determination of neighbors in the diagonal direction.

Finding a node's neighbor in a specified horizontal or vertical direction requires us to traverse ancestor links until a common ancestor of the two nodes is found. Once the common ancestor is located, we descend along a path that retraces the previous path with the modification that each step is a reflection of the corresponding prior step about the axis formed by the common boundary between the two nodes. For example, when attempting to locate the eastern neighbor of node L (i.e., node 0) in Fig. 1, node 1 is the common ancestor, and the eastern edge of the block corresponding to node L is the common boundary. During this process, encoded by procedure GTEQUAL_ADJ_NEIGHBOR, we pass through nodes 8, 2, 1, 3, and 0 in order. The expression of this procedure is facilitated by the predicate ADJ and the function REFLECT. ADJ(B, I) is true if and only if quadrant I is adjacent to boundary B of the node's block, e.g., ADJ(W, SW) is true. REFLECT(B, I) yields the SONTYPE value of the block of equal size that is adjacent to side B of a block having SONTYPE value I, e.g., REFLECT(N, SW) = NW, REFLECT(E, SW) = SE, REFLECT(S, SW) = NW, and REFLECT(W, SW) = SE.

Finding neighbors in a diagonal direction is more complicated. Once again, we must traverse ancestor links until a common ancestor of the two nodes is found. We require a three-step process. First, we locate the given node's nearest ancestor, say P, who is also adjacent (horizontally or vertically) to an ancestor of the sought neighbor, say Q. Next, we make use of GTEQUAL_ADJ_NEIGHBOR to locate Q. Finally, we retrace the remainder of the path while making directly opposite moves (e.g., a SW move becomes a NE move). The nearest ancestor of the first step is the first ancestor which is not reached by a link equal to the direction of the desired neighbor, e.g., to find a NW neighbor, the nearest such ancestor is the first ancestor node which is not reached via its son in the NW direction. For example, the NE neighbor of node F in Fig. 1 is EE. It is located by ascending the tree until the nearest ancestor 10, which is also adjacent horizontally (in this case) to an ancestor of EE, i.e., 11, is found. This requires going through a SE link to reach 10. Node 11 is now reached by applying GTEQUAL_ADJ_NEIGHBOR in the direction of the adjacency (i.e., east). This forces us to go through a NE link to reach 7 and a SW link to reach 2. Backtracking results in descending a SE link to reach 8 and a NW link to reach 11. Finally, we backtrack along the remainder of the path making 180^0 moves, i.e., we descend a NW link to reach EE. The expression of this is aided by the function COMMONSIDE ($Q1$, $Q2$) which indicates the boundary of the block containing quadrants $Q1$ and $Q2$ that is common to them (if $Q1$ and $Q2$ are not adjacent brother quadrants, then the value of COMMONSIDE is undefined). For example, COMMONSIDE (SE, SW) = S, while COMMONSIDE (NW, SE) is undefined.

V. Algorithm

The chessboard distance transform algorithm traverses the quadtree in postorder (i.e., the sons of a node are visited first). Recalling the definition of a neighbor given in Section III, we have that for each BLACK node of size 2^S its eight neighbors in the N, NE, E, SE, S, SW, W, and NW directions may have to be explored in determining the nearest WHITE node. If any of the neighbors is WHITE, then the minimum distance is 2^{S-1} and we cease further processing. Neighboring BLACK nodes do not affect the value of the chessboard distance transform since they result in a minimum transform value of $3 \cdot 2^{S-1}$ which exceeds the theoretical maximum. Thus, the heart of the algorithm lies in processing GRAY nodes.

The main procedure is termed CHESSBOARD_DIST and is invoked with a pointer to the root of the quadtree representing the image and an integer corresponding to the log of the diameter of the image (e.g., n for a $2^n \times 2^n$ image array). CHESSBOARD_DIST traverses the tree in postorder and controls the exploration of the eight neighbors of each BLACK node.

GTEQUAL_ADJ_NEIGHBOR locates a neighbor along a specified horizontal or vertical direction (e.g., N, E, S, or W). If the node is on the edge of the image, then no neighbor exists in the specified direction and NULL is returned [e.g., the western neighbor of node C is Fig. 1(b)]. If the node is not on the edge of the image and no neighboring BLACK or WHITE node exists, then a pointer to a GRAY node of equal size is returned [e.g., the eastern neighbor of node 1 in Fig. 3(a)]. In such a case, procedure DIST_ADJACENT continues the search by examining the subquadrants of the adjacent GRAY node. We first examine the nodes corresponding to the subquadrants adjacent to the side of the node being processed [e.g., subquadrants NW and SW of the eastern neighbor of node 1 in Fig. 3(a)]. If either node is WHITE, then a closest WHITE node in the specified direction has been found. If both nodes are GRAY, then we recursively apply DIST_ADJACENT to the corresponding subquadrants. If both nodes are BLACK, then we examine the remaining two subquadrants in a similar manner [e.g., subquadrants NE and SE of the eastern neighbor of node 1 in Fig. 3(a)].

GTEQUAL_CORNER_NEIGHBOR locates a neighbor adjacent to a specified corner (e.g., NE, SE, SW, or NW). If the node is on the edge of the image, then no neighbor exists in the specified corner and NULL is returned [e.g., the NW neighbor of node C in Fig. 1(b)]. If the node is not on the edge of the image and no neighboring BLACK or WHITE node exists, then a pointer to a GRAY node of equal size is returned [e.g., the NE neighbor of node 1 in Fig. 3(a)]. In such a case, procedure DIST_CORNER continues the search by examining the subquadrants of the adjacent GRAY node. We first examine the nodes corresponding to the subquadrant which is adjacent to the corner of the node being processed [e.g., subquadrant SW of the NE neighbor of node 1 in Fig. 3(a)]. If the node is WHITE, then a closest node in the specified direction has been found. If the node is GRAY, then we recursively apply DIST_CORNER to the subquadrant. If the node is BLACK, then we recursively examine the three remaining subquadrants in the following manner. We apply DIST_ADJACENT to the BLACK node's adjacent subquadrants in a direction which is adjacent to the BLACK node [e.g., DIST_ADJACENT is applied to the NW and SE subquadrants of the NE neighbor of node 1 in Fig. 3(a)]. We also apply DIST_CORNER to the nonadjacent subquadrant [e.g., the NE subquadrant of the NE neighbor of node 1 in Fig. 3(a)].

As an example of the application of the algorithm, consider the region given in Fig. 1(a). Fig. 1(b) is the corresponding block decomposition, while Fig. 1(c) is its quadtree representation. All of the BLACK nodes have labels ranging from A to R, while the WHITE nodes have labels ranging between AA and PP. The GRAY nodes have labels ranging between 1 and 11. The BLACK nodes are labeled in the order in which their adjacencies are explored by CHESSBOARD_DIST. Fig. 1(d) contains the chessboard distance transform corresponding to Fig. 1(b).

```
procedure CHESSBOARD_DIST(P, LEVEL);
/* Given a quadtree rooted at node P spanning a 2↑LEVEL by
   2↑LEVEL space, find the Chessboard distance of each BLACK
   node to its closest WHITE node. WHITE nodes are assigned
   distance ϕ */
begin
   value node P;
   node Q;
   value integer LEVEL;
   integer C;
   quadrant I;
   direction D;
   if GRAY(P) then
```

```
        begin
          for I in {"NW," "NE," "SW," "SE"} do
              CHESSBOARD_DIST(SON(P,I),LEVEL-1);
        end
      else if BLACK(P) then
        begin
          D←"N";
          C←2↑LEVEL;
          do
            begin
              Q←GTEQUAL_ADJ_NEIGHBOR(P,D);
              D←if NULL(Q) or BLACK(Q) then C
                else if WHITE(Q) then ∅
                else DIST_ADJACENT(Q,QUAD(OPSIDE(D),
                        CCSIDE(D)),QUAD(OPSIDE(D),
                        CSIDE(D)),2↑(LEVEL-1),∅,C);
              if C≠0 then
                begin
                  Q←GTEQUAL_CORNER_NEIGHBOR(P,
                        QUAD(D,CSIDE(D)));
                  D←if NULL(Q) or BLACK(Q) then C
                    else if WHITE(Q) then ∅
                    else DIST_CORNER(Q,D,CSIDE(D),
                        2↑(LEVEL-1),∅,C);
                  D←CSIDE(D);
                end;
            end
            until C=∅ or D="N";
          DIST(P)←C+2↑(LEVEL-1);
        end
      else DIST(P)←∅; /* a WHITE node */
end;

node procedure GTEQUAL_ADJ_NEIGHBOR(P,D);
/* Return the neighbor of node P in horizontal or vertical
   direction D. If such a node does not exist, then return
   NULL */
begin
   value node P;
   node Q;
   value direction D;
   if not NULL(FATHER(P)) and ADJ(D,SONTYPE(P)) then
        /* Find a common ancestor */
      Q←GTEQUAL_ADJ_NEIGHBOR(FATHER(P),D)
   else Q←FATHER(P);
   /* Follow the reflected path to locate the neighbor */
   return (if not NULL(Q) and GRAY(Q) then SON(Q,
          REFLECT(D,SONTYPE(P)))
          else Q);
end;

integer procedure DIST_ADJACENT(P,Q1,Q2,W,B,C);
/* Given a subquadtree rooted at node P spanning a 2 · W X
   2 · W space, the distance of the closest WHITE node to the
   border formed by quadrants Q1 and Q2 of P. B is a lower
   bound for the distance. C is the minimum distance ob-
   tained so far */
begin
   value node P;
   value quadrant Q1,Q2;
   value integer B,C,W;
   return (if B≥C then C /* The minimum has already been
                            found */
      else if WHITE(P) then B
      else if BLACK(P) then C
      else if BLACK(SON(P,Q1)) and BLACK(SON(P,Q2)) then
          DIST_ADJACENT(SON(P,OPQUAD(Q1)),Q1,Q2,
                W/2,B+W,
              DIST_ADJACENT(SON(P,
                OPQUAD(Q2)),Q1,Q2,
                W/2,B+W,C))
      else DIST_ADJACENT(SON(P,Q2),Q1,Q2,W/2,B,
              DIST_ADJACENT(SON(P,Q1),
                Q1,Q2,W/2,B,C)));
end;

node procedure GTEQUAL_CORNER_NEIGHBOR(P,C);
/* Return the neighbor of node P in diagonal direction C. If
   such a node does not exist, then return NULL */
begin
   value node P;
   node Q;
   value quadrant C;
   if not NULL(FATHER(P)) and SONTYPE(P)≠OPQUAD(C)
      then if SONTYPE(P)=C then Q←GTEQUAL_CORNER_
      NEIGHBOR(FATHER(P),C) else Q←GTEQUAL_ADJ_
      NEIGHBOR(FATHER(P),COMMONSIDE(SONTYPE(P),C))
   else Q←FATHER(P);
   /* Follow the opposite path to locate the neighbor */
   return (if not NULL(Q) and GRAY(Q) then SON(Q,
          OPQUAD(SONTYPE(P)))
          else Q);
end;

integer procedure DIST_CORNER(P,D1,D2,W,B,C);
/* Given a subquadtree rooted at node P spanning a 2 · W X
   2 · W space, return the distance of the closest WHITE node
   to the corner formed by quadrant OPQUAD(QUAD(D1,D2))
   of P. B is a lower bound for the distance. C is the mini-
   mum distance obtained so far. */
begin
   value node P;
   value direction D1,D2;
   value integer B,C,W;
   integer TEMP;
   if C ≥ D then return (C) /* The minimum has already been
                               found */
   else if WHITE(P) then return (B)
   else if BLACK(P) then return (C)
   else
      begin
        TEMP←DIST_CORNER(SON(P,OPQUAD(QUAD(D1,
             D2))),D1,D2,W/2,B,C);
        TEMP←DIST_ADJACENT(SON(P,QUAD(D1,
                OPSIDE(D2))),
              OPQUAD(QUAD(D1,D2)),
              QUAD(OPSIDE(D1),D2),
                W/2,B+W,TEMP);
        TEMP←DIST_ADJACENT(SON(P,QUAD(OPSIDE(D1),
                D2)),
              QUAD(D1,OPSIDE(D2)),
              OPQUAD(QUAD(D1,D2)),
                W/2,B+W,TEMP),
        TEMP←DIST_CORNER(SON(P,QUAD(D1,D2),D1,D2,
                W/2,B+W,TEMP);
        return (TEMP);
      end;
end;
```

VI. Concluding Remarks

The concept of distance has been applied to images represented by quadtrees. A related concept, termed a chessboard distance transform, has been defined and an algorithm has