COMPUTER VISION, GRAPHICS, AND IMAGE PROCESSING 26, 207–216 (1984)

Octree Representations of Moving Objects*

NARENDRA AHUJA AND CHARLES NASH[†]

Coordinated Science Laboratory and Department of Electrical Engineering, University of Illinois, 1101 W. Springfield Avenue, Urbana, Illinois 61801

Received July 26, 1982; revised December 30, 1982; accepted February 7, 1983

An algorithm is described that updates an object's octree representation as the object is linearly translated through space. This is accomplished by performing simple arithmetic on the path representations of the nodes to be translated. Among others, one advantage of the algorithm is in devising collision-free and efficient trajectories of moving objects in robotics.

1. INTRODUCTION

An important class of representations of three dimensional objects is based upon recursive decomposition of space until each element of the decomposition is completely within or outside the object of concern, or the limit of resolution is reached. A common method involves cubical regions of space and their decomposition into octants. This gives an octree description of the occupancy of space by objects [1, 7, 12, 13]. A given tree is defined for a certain choice of the origin and the orientation of the coordinate axes. Different choices of the locations of the axes give drastically different trees because the objects get decomposed differently. A slight shift in an object's location may result in a much larger or more compact tree. This is the major disadvantage that decomposition-based representations have when compared to variable position block representations such as medial axis transforms [12] and piecewise approximation [2]. The latter representations do not change as an object is moved, but they may be computationally more expensive to obtain.

When dealing with a static environment, the octree description will need to be derived only once for a given choice of the axes. However, it is often necessary to represent scenes containing moving objects. For example, in robotic manipulation of environment, representations of dynamic scenes are necessary to design collision-free and efficient trajectories for object movement. Trees must be continuously obtained to reflect varying positions of objects with respect to fixed axes. Since the tree shapes are very sensitive to object locations [5], one approach is to track the moving objects or compute their positions continuously, and rederive tree representations for each configuration of interest. For applications such as collision avoidance, tests must be made fairly often to detect potential collisions and to take appropriate measures to

*The support of the Joint Services Electronics Program (U.S. Army, Navy, and Air Force) under Contract N00014-79-C-0424 and the help of Monica Ray in preparing this paper are gratefully acknowledged.

[†]Current address: Cincinnati Milacron, Mason-Morrow Road, Lebanon, Ohio 45036.

avert them. The exact frequency of the tests is determined by the motion parameters and must be sufficiently high to allow only limited changes in the object configurations between successive test instants. Thus, successive configurations are closely related, and hence, so must be their octrees, despite significant changes in the octree shapes. This interdependence may be exploited to save some computation in obtaining successive trees by updating current trees instead of rederiving them completely. This paper describes an algorithm for updating an octree as the represented object undergoes translation.

Section 2 reviews the octree representation of three-dimensional objects. Section 3 describes an algorithm to update the octree of an object as the object undergoes translation. The algorithm updates the octree by performing simple arithmetic on the path representations of the nodes to be translated. Section 4 summarizes the experimental results. Section 5 presents concluding remarks.

2. OCTREES

The octree representation [1, 6, 7, 11, 12, 13] of the occupancy of space by objects is obtained by recursive decomposition of the space into octants (Fig. 1). To obtain the octree for a given object, we start with the entire space as a single (starting) block. If the block under consideration is completely contained within the object it is left alone; otherwise, it is divided into eight octants (Figs. 1, 2a) each of which is treated similarly. The splitting continues until all the blocks are either completely within or completely outside the object (Fig. 3a), or a block of minimum allowed size (representing the finest resolution) is reached.

Such recursive subdivision (Fig. 2b) allows a tree description (Fig. 2c) of the occupancy of space. Each block corresponds to a node in the tree. Let us label a leaf black or white if it corresponds, respectively, to a block which is completely contained within the object or within the free space. Nonleaf nodes are labeled gray, and have children unless they are of the minimum allowed size, in which case they

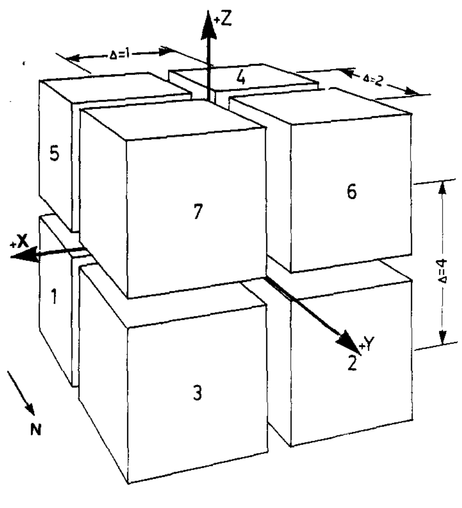

FIG. 1. A block and its decomposition into octants. The south-west, south-east, north-west, and north-east octants in the lower layers are labeled 0, 1, 2, and 3, respectively. The corresponding octants in the upper layer are labeled 4, 5, 6, and 7.

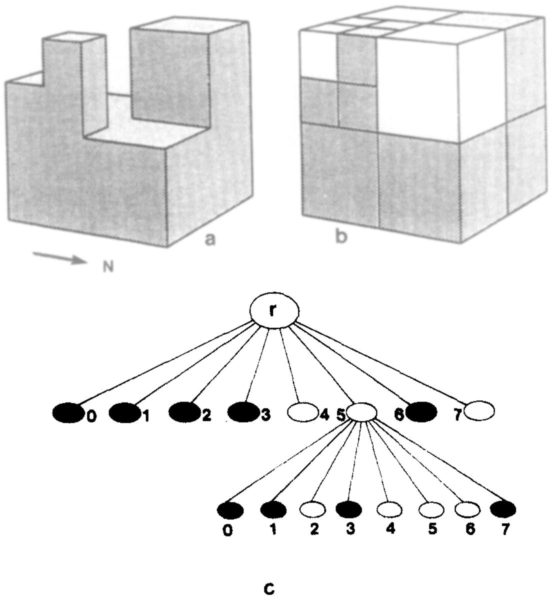

FIG. 2. (a) A simple object. (b) Recursive subdivision of space superimposed on the object in (a) to obtain its representation. (c) Octree for the object in (a); r denotes the root node. Black (white) leaves are indicated by darkened (empty) circles.

are relabeled black. The free space covered by such nodes is thus treated as part of the object in order to be conservative in detecting interference.

The root (level 0) of the octree represents the largest sized block (the overall representation space). Successively lower levels represent blocks whose side lengths halve with each lower level. The separation between adjacent blocks at a given level also halves for each lower level. The size of the smallest block is limited by resolution. We will denote the side length of the smallest block by unity. Then the total number of possible levels L is $\log_2$ (side length of largest block). The leaves of the octree may occur at many levels. The octree may have fewer than L levels if the object can be represented by blocks of side lengths larger than unity.

An equivalent but more economical and simpler representation is obtained if only the object volume is explicitly represented, and the remaining space is assumed to be

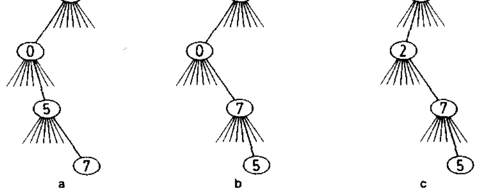

FIG. 3. Translating a unit size block parallel to the positive y axis. The links not leading to any node denote "NIL" links or white children. (a) A unit-size object block $r057$. (b) The object in (a) translated by a unit distance. (c) The object in (a) translated by 5 distance units.

free. Thus, only the black leaves in the octree are retained. The white children of a gray node are deleted and their corresponding links in the parent are marked distinctly, say pointing to "NIL." Thus in Fig. 2c the leaves which are empty circles can be deleted. In this paper we will use such a tree representation to reduce the storage space and the time taken by the tree-updating algorithms.

3. TRANSLATION

Given the translational motion parameters of an object and the time interval after which its octree must be updated, the object displacements along the three orthogonal (x, y, z) axes can be computed. In this section we describe an algorithm that receives the octree of an object and the displacement triple, and provides the octree of the translated object. The three axial components of the translation vector are used sequentially to move the nodes in the tree to their new positions, and the tree is compacted before the algorithm terminates. Each of the translational components is assumed to be an integral distance, a multiple of the unit distance representing the smallest block's dimensions.

3.1 Translating a Unit Block

Consider moving a block of unit size a unit distance parallel to the positive x axis. In general, the blocks in the initial representation corresponding to octants in the western half of a nonleaf node will move to occupy adjacent octants in the eastern half (Fig. 1). The corresponding nodes in the tree, having labels 0, 2, 4, and 6, will acquire the labels 1, 3, 5, and 7, respectively (Fig. 1). Thus the new labels are obtained by adding 1 to the old labels. Symmetrically, the nodes corresponding to octants in the eastern half of a node, having labels 1, 3, 5, and 7, will change their labels to 0, 2, 4, and 6, respectively, corresponding to adjacent octants to the west. Thus the new labels are obtained by adding 7, modulo 8, to the old label. However, the new octants do not share the same parent node. The new parent block is that to the east of the original parent block. Thus the parent node in the tree must also move. This means that along with the label of the original node the label of its parent node also changes. Depending upon the position of the originally moved node in the tree, the label change may propagate to any height up the tree.

TABLE 1
Label Increments (Modulo 8) Representing Unit Displacements

Initial	Increment after unit displacement along		
label	$+X$	$+Y$	$+Z$
0	+1	+2	+4
1	+7	+2	+4
2	+1	+6	+4
3	+7	+6	+4
4	+1	+2	+4
5	+7	+2	+4
6	+1	+6	+4
7	+7	+6	+4

Translation parallel to the y axis differs from that parallel to the x axis in that the labels change, modulo 8, by 2 or 6 (Fig. 1), and the roles of east and west octants are played by south and north octants, respectively. Similarly, translation along the z axis changes, modulo 8, the labels by 4. Table 1 lists the required increments in the values of the labels of nodes when translated by a unit distance in the positive direction along all the three axes; Table 2 lists the corresponding label transitions. When along with a node its parent must also change labels, this is indicated in Table 2 by appending a "carry" (a 1) to the left of the new label of the node, meaning that the parent octant should also be moved to the adjacent octant in the direction of translation.

Consider, for example, the octree of a unit size block shown in Fig. 3a. Let us denote the leaf by the sequence $r057$ formed by appending to root r the labels encountered along the path leading from the root to the leaf. To move the leaf block by a unit distance parallel to the positive y axis, we perform the following addition (Table 2):

$$\begin{array}{r} r057 \\ \underline{001} \\ r075 \end{array}$$

Addition of 7 and 1 generates the label 5 and carry 1. Because of the carry, label 5 is once again updated using Table 2. The resulting tree is shown in Fig. 3b. Translation by more than a unit distance along a positive axis is performed as follows. The translation is first expressed as a binary integer. Since the dimensions of the blocks corresponding to nodes at successively higher levels double, successively more significant bits of the binary integer may be associated with translations by side lengths of nodes (or corresponding blocks) at successively higher levels. The least significant bit of the binary integer is associated with the lowest possible level whose nodes represent unit-size blocks. The position of the translated leaf node is given by the sum, according to Table 2, of the sequence $r** \cdots *$ denoting the original leaf and the binary integer representing the desired translation. As an example, suppose we want to move the block in Fig. 3a by 5 (binary 101) units parallel to the positive y axis. The following addition gives the octree after transla-

TABLE 2
Label Transitions Representing Unit Displacements

Initial	Final label after unit displacement along		
label	$+X$	$+Y$	$+Z$
0	1	2	4
1	10	3	5
2	3	10	6
3	12	11	7
4	5	6	10
5	14	7	11
6	7	14	12
7	16	15	13

tion (Fig. 3c):

$$\begin{array}{r} r057 \\ \underline{101} \\ r275 \end{array}$$

Translation of a leaf by $t, t > 0$, parallel to a negative axis is equivalent to a translation by $-t$ parallel to the corresponding positive axis. Since to obtain the octree a block is divided into two halves along each of the three axes, the various additions listed in Table 2 are binary additions, where the pairs of "bit" labels are chosen from among the integers 0 through 7. A negative translation is performed exactly as before but using the 2's complement representation of $-t$. For example, to translate $r275$ by 5 units parallel to the negative y direction, we perform a translation by -5 (2's complement 011) units parallel to the positive y axis, as follows:

$$\begin{array}{r} r275 \\ \underline{011} \\ r1057 \end{array}$$

carry lost

where the carry out of the most significant position is ignored. This gives the translated node—the original leaf node in the previous example.

3.2 Translating an Object

The leaf nodes of the octree of an arbitrary object represent blocks of various sizes constituting the object and their configuration. The leaves may occur at different levels. To translate the object some distance, we first obtain projections of the translation along the three axes. Each of these translations is then performed by executing the following steps. The displacement value is represented as a binary integer. Let H be the least significant bit position in the binary representation containing a 1. Each black leaf node in the tree is recursively assigned eight black children down to level H. Then the nodes at levels $\leq H$ are traversed in postorder such that at any given level, nodes in the direction of translation are encountered before those in the opposite direction. Thus, if the translation is positive in sign, the traversal is postorder and from right to left; all eight children of a node are traversed right to left (in order of decreasing labels) before the node itself is traversed. Conversely, if the displacement is negative in sign the traversal is postorder and from left to right. Due to the notation used for labeling a node's children, positive displacements always shift nodes to the right and negative displacements always shift nodes to the left. As the tree is traversed, each level H node is translated as follows.

First, using Table 2, the label sequence, say rs', describing the destination location after translation of the given node is computed from the position sequence rs of the node known from the traversal algorithm, and the binary integer representing the displacement. The destination location is then accessed, starting from the root of the tree and moving down according to s'. This may require adding new branches and nodes when the path s' does not already exist, e.g., when the last node is translated. Finally, the data (color, children links) of the source node is copied into the destination node, and the original node is marked gray with no children. (This node represents currently free space. If no object blocks move into the space represented by this node, i.e., it remains gray with no children, at the end of translation, it will be deleted during the tree compaction phase before the algorithm terminates.)

When the traversal is complete all the leaf nodes have been translated. One could say that the program metaphorically swings through the octree, encounters nodes to be shifted, and "throws" each such node behind its own path of progression to perform the shift. At any point during the traversal the nodes ahead of the point of procession have not yet been translated and nodes behind the point of procession have all been translated. As the final step, the tree is traversed again in postorder to delete superfluous nodes. Each gray node having no children, as well as the corresponding child link of its parent node, are deleted. Black siblings are deleted and their parent marked as a black leaf. The resulting octree thus contains only black leaf nodes.

This algorithm was implemented in PASCAL on a PDP-11/40. Note that the algorithm allows only translation by an integer (i.e., integral multiple of smallest block size). Nonintegral translations or translations by distances smaller than the unit resolution cannot be performed exactly. To do so, either the resolution must be increased, or the magnitude of the translation rounded off to the nearest integer to obtain an approximate representation.

3.3 Analysis

The translation algorithm described above consists of three major steps performed on the octree obtained by adding black nodes as described in Section 3.2.

(1) Traverse the tree levels $\leq H$.
(2) For each level H node encountered in (1):
  (i) Compute the destination location.
  (ii) Access the destination node. If it does not already exist, insert the necessary node sequence in the tree.
  (iii) Copy the source node data into the destination node.
  (iv) Modify data in the source node (mark it gray with no children).
(3) Traverse the tree again for compaction.

Consider step 2. Substeps 2i, 2iii, and 2iv each requires $O(1)$ number of operations. Step 2ii, however, must follow a path from root to level H, performing $O(1)$ operations at each node along the path. Thus, this substep requires $O(H)$ operations.

Step 1 requires a number of operations proportional to the number of nodes in the octree at levels $\leq H$. The traversal in step 3 involves an octree that, in general, contains more nodes than the tree in step 1, because of the presence of leftover gray nodes. $O(1)$ operations are performed at each node. Thus, the total number of operations required in step 3 is O(number of nodes in the tree at levels $\leq H$).

Let N_i and N_l denote the total number of nodes at levels $\leq H$ and H, respectively, in the tree in step 1. Let N' denote the number of tree nodes in step 3.

```
a) 0, GREY
      7, GREY
         0, GREY
            0, GREY
               2, GREY
                  0, GREY
                     2, BLACK
                     6, GREY
                        0, GREY
                           0, BLACK
                           1, BLACK
                           2, BLACK
                           3, BLACK
                        1, GREY
                           0, BLACK
                           1, BLACK
                           2, BLACK
                           3, BLACK
                        2, GREY
                           0, BLACK
                           1, BLACK
                           2, BLACK
                           3, BLACK
                        3, GREY
                           0, BLACK
                           1, BLACK
                           2, BLACK
                           3, BLACK
```

```
b) 0, GREY
      5, GREY
         2, GREY
            0, GREY
               0, GREY
                  0, GREY
                     1, GREY
                        2, BLACK
                        3, BLACK
                        6, BLACK
                        7, BLACK
                     5, GHEY
                        2, GREY
                           0, BLACK
                           1, BLACK
                           2, BLACK
                           3, BLACK
                        3, GREY
                           0, BLACK
                           1, BLACK
                           2, BLACK
                           3, BLACK
               2, GREY
                  2, GREY
                     3, GREY
                        0, BLACK
                        1, BLACK
                        4, BLACK
                        5, BLACK
                     7, GREY
                        0, GREY
                           0, BLACK
                           1, BLACK
                           2, BLACK
                           3, BLACK
                        1, GREY
                           0, BLACK
                           1, BLACK
                           2, BLACK
                           3, BLACK
```

FIG. 4. Octree map output of the translation algorithm. Nodes not shown are white. (a) The original object. (b) The object in (a) translated by $(4, -50, 0)$.

Then the total number of operations needed to perform the translation is on the order of

$$\begin{aligned} O(N_i + N_l) + O(N_l H) + O(N') &= O(N_l H) + O(N') \\ &= O(N_l H + N'). \end{aligned}$$

4. EXPERIMENTAL RESULTS

The algorithm given in Section 3 was implemented in PASCAL on a PDP-11/40. A program was written to generate octrees from face descriptions of blocky objects having faces parallel to the coordinate planes. The translation algorithm was then applied to the resulting octrees. The octrees were output as indented node lists instead of tree structures for simplicity. Figure 4a shows the node lists of the octree of a simple parallelopiped. Figure 4b shows the original tree after translation.

5. CONCLUSIONS

A major advantage of the spatial decomposition based representations is the ease with which various parts of space can be accessed and knowing if they contain parts of the objects represented. In this respect they differ from maximal block-fit-based representations [2, 12], in which occupancy of a given region must be determined by conducting a search over the blocks comprising the representation. However, the latter have the advantage that object movement is trivial to perform since the representation explicitly lists all blocks. The decomposition-based representations build associations between object segments and cells in the decomposition, thus splitting the object and making object manipulation more cumbersome and obscure. Translation and rotation [7] algorithms help to relate the node configuration in the octree to the object structure to simplify object manipulation, while still retaining the explicit representation of space occupancy. Algorithms similar to those for octrees can be obtained for updating quadtrees of two-dimensional moving objects or image regions. The need for translation and rotation algorithms for image regions from their regular decomposition was pointed out by Klinger and Dyer [8].

Our motivation for the work reported here came from the problem of collision avoidance among objects in robotics. Detecting interference between two objects is easy if octree representations of the objects are given, since it requires a single parallel traversal of the two trees [1]. However, for moving objects it is essential to have efficient updating procedures for keeping the representation current. One octree is maintained for the entire static environment and one is maintained for each of the moving objects. Given translation, rotation, and other geometric manipulation algorithms, the collision avoidance methods based on octrees outlined in [1] should be feasible. We have done some preliminary work on planning collision-free and efficient trajectories for moving an object to a given destination.

A rotation by integral multiples of 90° does not change the number of nodes in the tree, and the nodes in the new tree have a one-to-one correspondence with those in the tree before rotation. Jackins and Tanimoto [7] present an algorithm to update the octree after rotation. They give the relationship between a node's labels before and after rotation by integral multiples of 90°. If arbitrary rotations are allowed, the number of blocks required to represent an object may become excessively large. Moreover, the trend towards fragmentation may be persistent even when symmetric pairs of rotations are performed. This is because when a compactly represented

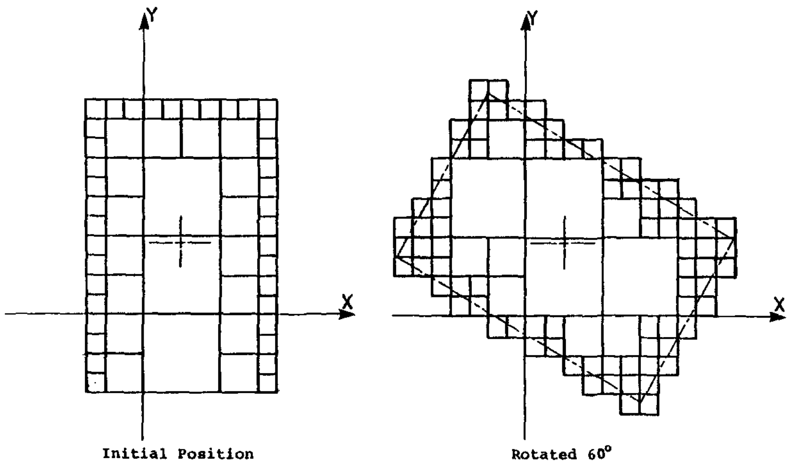

FIG. 5. Block fragmentation due to rotation of a two-dimensional object. (a) Initial blocks. (b) Blocks after rotation.

object is rotated by an arbitrary angle, it may be representable only approximately by, say, a staircase of relatively small blocks (Fig. 5). A pure inverse rotation will not restore the tree to the original shape because of the approximations made in the original rotation, unless the shapes of the objects represented by the tree are monitored and any approximation errors are detected and corrected continuously. These latter procedures may be computationally too expensive to retain the edge the octree representation may have over more direct surface or volume based representations [3, 4, 9, 10, 14]. The problems encountered in implementing arbitrary rotations are also faced in representing objects having complex shapes.

REFERENCES

1. N. Ahuja, R. T. Chien, R. Yen, and N. Bridwell, Interference detection and collision avoidance among three dimensional objects, Proc. 1st National Conf. on Artificial Intelligence, Stanford University, August 19–21, 1980, 44–48.
2. N. Ahuja, L. S. Davis, D. L. Milgram, and A. Rosenfeld, Piecewise approximation of pictures using maximal neighborhoods, *IEEE Trans. Comput.* **C-27**, 1978, 375–379.
3. J. W. Boyse, Interference detection among solids and surfaces, *Commun. ACM* **22**, 1979, 3–9.
4. P. G. Comba, A procedure for detecting intersections of three-dimensional objects, *J. ACM* **15**, 1968, 354–366.
5. M. Li, W. I. Grosky, and R. Jain, Normalized quadtrees with respect to translations, Proc. IEEE Computer Society Conf. on Pattern Recognition and Image Processing, Dallas, August 3–5, 1981, 60–62.
6. G. M. Hunter, Efficient computations and data structures for graphics, Ph.D. dissertation, Electrical Engineering and Computer Science Department, Princeton University, June 1978.
7. C. L. Jackins and S. L. Tanimoto, Oct-trees and their use in representing 3D objects, *Computer Graphics and Image Processing*, **14**, 1980, 249–270.
8. A. Klinger and C. R. Dyer, Experiments in picture representation using regular decomposition, *Computer Graphics and Image Processing*, **5**, 1975, 68–105.
9. T. Lozano-Perez and M. A. Wesley, An algorithm for planning collision-free paths among polyhedral obstacles, *Commun. ACM* **22**, 1979, 560–570.
10. K. Maruyama, A procedure for detecting intersections and its application, University of Illinois Computer Science Technical Report No. 449, May 1971.
11. D. J. Meagher, Efficient synthetic image generation of arbitrary 3-d objects, Proc. IEEE Computer Society Conf. on Pattern Recognition and Image Processing, Las Vegas, June 14–17, 1982, 473–478.
12. A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, Academic Press, New York, 1982.
13. S. N. Srihari, Hierarchical representations for serial section images, Proc. 5th Int. Conf. on Pattern Recognition, 1980, 1075–1080.
14. S. Udupa, Collision detection and avoidance in computer controlled manipulators, Proc. 5th Int. Joint Conf. Artificial Intelligence, Cambridge, Massachusetts, 1977, 737–748.