The Determination of Next Best Views

C. I. Connolly
(ARPANet address: connolly@GE)

Computer Science Department
Rensselaer Polytechnic Institute

ABSTRACT

There are situations in which one would like to know a good sequence of range-image views for obtaining a complete model of a scene. This paper describes two algorithms which use partial octree models to determine the "best" next view to take.

1. Introduction

In certain situations, complete models of a scene are desirable. In order to obtain a complete model, one must select a set of views which, in some sense, "covers" the scene. One should also choose a modeling scheme from which such covering information can be easily extracted. This paper will describe algorithms for finding such a set of covering views using the octree data structure. The views are represented by range images of the scene.

Previous work ([1], [2], [3], [4]) has addressed the assimilation of range data into octree models. In addition to the advantages of octrees which were discussed in those papers, another advantage becomes apparent when considering the notion of finding covering views (view completion). Namely, octrees apply a label to all of space within the octree cube ([5]). This feature makes it relatively easy to find what is, in some sense, the next "best" view.

Two algorithms are discussed in this paper. The first is the most obvious and simple method, and is presented for comparison with the second algorithm, which does not deal with self-occluding scenes as well, but whose implementation is much faster and simpler. In both cases, the algorithms assume that a partial octree model is the input. In addition, they make the simplifying assumption that for all views, the sensor points toward the origin. In this paper, the term "view vector" will refer to the position of the sensor.

2. The Octree Structure

Octrees are 8-ary trees which recursively divide a cube into eight subcubes. The octree structure used herein used contains four node types. *Parent* nodes are those which have children. Leaves can belong to one of three types: *Empty*, *Occupied*, and *Unseen*. Nodes which are marked *Empty* consist of empty space which is visible from at least one of the views used to generate the octree. Nodes which are *Occupied* contain points on the object's surface, and nodes which are *Unseen* are those volumes which have not yet been seen by any view. The root of the octree is always initialized to the *Unseen* state before views are assimilated. A method for the assimilation of arbitrarily oriented range views into octrees is given in [4].

With this structure and an algorithm which produces a single view vector indicating the next location for a sensor, the following steps are used to complete the octree model:

1) Pick an arbitrary view vector to start with.
2) Assimilate the most recently selected view into the octree.
3) Determine the next view vector.
4) If no such vector exists, terminate, else go back to 2)

3. The Planetarium Algorithm

The first algorithm presented here sets up a sphere around the scene. This sphere is evenly sampled along latitude and longitude. At each sample point, a hidden-line view of the octree is generated considering *only Occupied and Unseen nodes.* Following this, the area of *Unseen* nodes covering the viewplane is determined. This value provides a measure of how much of the unknown area can be eliminated by taking a view from each point, and takes into account occlusion by other *Unseen* and *Occupied* nodes. (See figure 1)

To obtain a set of views which will complete (as well as the sphere sampling allows) the octree, one can use steps 1-4 above. The point on the sphere with the largest *Unseen* area is selected as the view vector for the next view. Naturally, the octree may have gaps if, for instance, the sphere sampling resolution was too small.

This algorithm usually terminates after taking 4 or 5 views. (See figure 2) The objects considered were turbine blades and simple polyhedral scenes. As may be obvious from the above, the algorithm is not very economical in terms of time, but does tend to select a small number of views. It also can be considered optimal in the sense that, given a dense sphere sampling, it will find that view which will eliminate the largest *Unseen* volume.

4. The Normal Algorithm

The second algorithm uses information about the faces in an octree which are common to both *Unseen* and *Empty* nodes. Such faces will be referred to as *Unseen/Empty* faces. In effect, this algorithm measures the visibility of the *Unseen* nodes within the tree. The algorithm determines the number of faces of each *Unseen* node that are exposed to the region of space which is known to be empty. This is accomplished by recording the areas of *Unseen/Empty* faces. Six separate area sums are kept, since the faces can be oriented in any of six directions.

To explain this more precisely, the following is a description of the steps involved in the algorithm:

```
For each node $N$ of the octree:
    If $N$ is an Unseen node, then
        For i from 1 to 6:
            If $neighbor_i$ is Empty, then
            $count_i = count_i + 4^{-j}$ where j
            is the level of node $N$.
            else if $neighbor_i$ is a Parent
            node, then
                for each descendant, $M$,
                of $neighbor_i$ which abuts
                the original node $N$:
                    if $M$ is Empty, then
                    $count_i = count_i + 4^{-k}$,
                    where k is the level
                    of node $M$.
```

Note that if the neighbor of an *Unseen* node is a *Parent* node, only those subnodes of it which are adjacent to the *Unseen* node are examined (See figure 4).

Finding neighbors is accomplished by manipulating tree addresses. Associated with the address for each node is a level number l and 3 bit strings, each of length l, which describe traversal of the tree in each of the x, y, and z directions. To compute neighbor addresses in a given direction, the number N represented by the corresponding bit string is incremented or decremented. An out-of-bounds condition occurs when the actual bit string length exceeds the level number, i.e., when $N > 2^l - 1$ or $N < 0$.

What ultimately results in each $count_i$ is the area of the *Unseen/Empty* faces in each of six directions. Each $count_i$ can be thought of as corresponding to a face of a cube. Keeping this in mind, the next view direction is found by locating the cube vertex whose three adjoining faces have the highest values. Each $count_i$ corresponds to a coordinate axis, i.e., x, $-x$, y, $-y$, z, and $-z$. To obtain the view direction vector, the $count_i$ with the largest magnitudes for each axis are selected. This prevents regions of *Unseen/Empty* faces from cancelling each other out. Thus, the three maxima form a direction vector which is the direction from the origin in which the sensor is to be placed for the next view (See figure 4). The next position of the sensor should lie anywhere on a ray in the direction of the vector emanating from the origin. In practice, it suffices to compute the new sensor position as the vector scaled to some constant radius from the origin.

By way of comparison, the Planetarium algorithm normally takes on the order of 30 minutes of (VAX 11/780) CPU time to determine the view vector. The Normal algorithm generates a view vector in less than 30 CPU seconds.

5. Conclusion

Two algorithms have been provided for determining the "best" direction for placing a sensor in order to gain the most information about a scene. Both algorithms assume that the sensor points toward the origin. The second algorithm is much faster than the first, but will not deal with occlusions as well as the first. For example, it is quite possible that a set of *Unseen/Empty* faces yields a view for which that set is occluded, as in figure 5. Current research is aimed at determining oblique views which would circumvent this problem.

In the above algorithms, no sensor position was prohibited. In practical situations though, it is usually the case that the sensor cannot move freely. This represents a collision avoidance problem. The above algorithms would therefore need to be modified to consider only those view vectors which lie within the volumes that are known to be *Empty*. In addition, it might be desirable to assume that, after the first view, the space outside the bounds of the octree (the void) is prohibited as well. The current implementations of the above algorithms assume that the void is *Empty*. Alternately, vectors which fall within prohibited areas could be forced toward *Empty* space.

Also being examined is the use of operators on octrees ([6]). It is possible that curvature (or other) operators will yield enough information about the structure of the scene represented by the octree to segment the octree. This could allow the ability to isolate distinct components of the scene.

Acknowledgements should go to Joe Mundy and Ross Stenstrom, both of whom made major contributions to the formulations of these algorithms.

6. References

[1] M. Schneier, E. Kent, and P. Mansbach, *Representing Workspace and Model Knowledge for a Robot with Mobile Sensors,* 7th Int'l Conf. on Pattern Recognition,

Montreal, 1984

[2] W. N. Martin and J. K. Aggarwal *Volumetric Descriptions of Objects from Multiple Views,* IEEE PAMI, March 1983, vol. 5, #2

[3] Mann-may Yau, S. N. Srihari, *A Hierarchical Data Structure for Multidimensional Digital Images,* CACM, July 1983, vol. 26, #7

[4] C. I. Connolly, *Cumulative Generation of Octree Models from Range Data,* Proc. 1st IEEE Int'l Robotics Conf., Atlanta, 1984

[5] J. L. Mundy, Personal Communication

[6] J. R. Stenstrom, *Finding Points of High Curvature and Critical Curves from a Volume Model,* Proc. 1985 IEEE Int'l Conference on Robotics and Automation, St. Louis, 1985

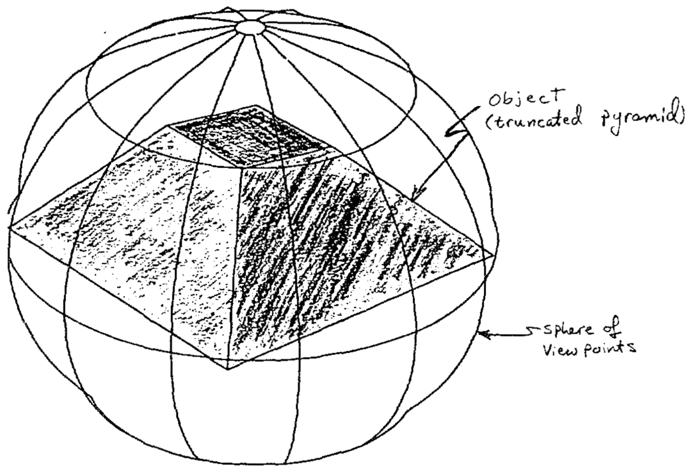

Figure 1. Sphere with point selection

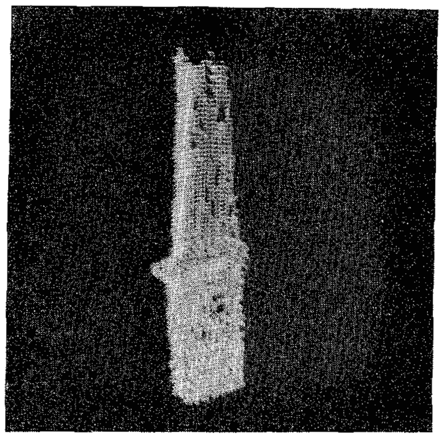

Figure 2. Completed octree

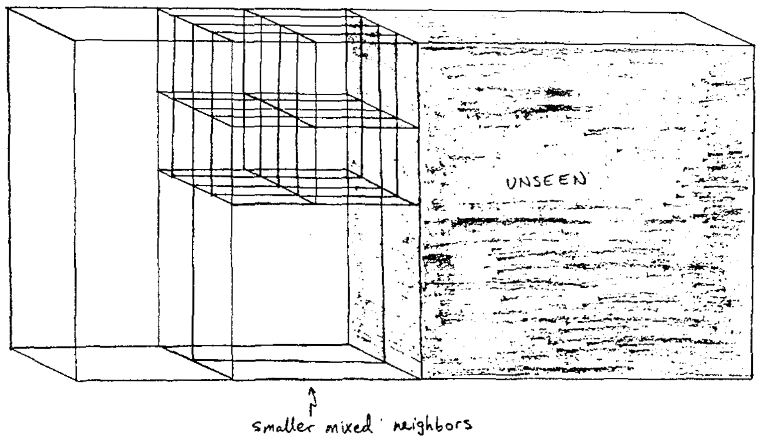

Figure 3. *Unseen* node with smaller *Empty* neighbors

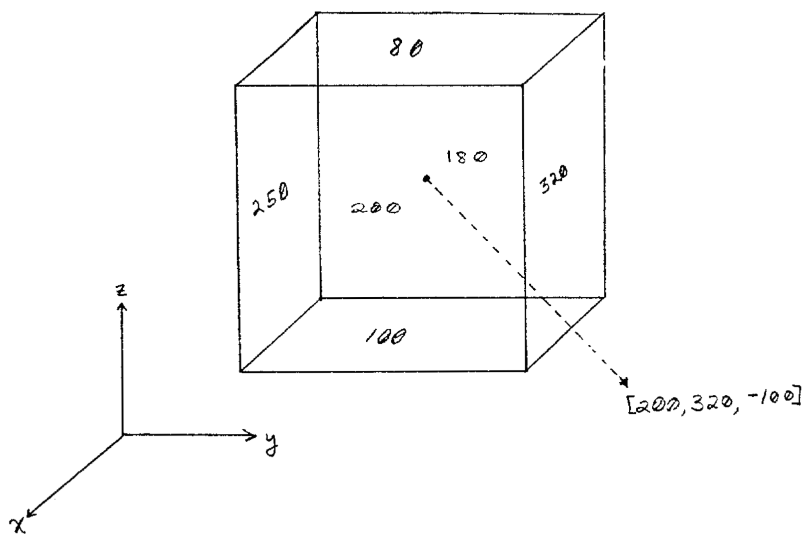

Figure 4. Computation of new view vector

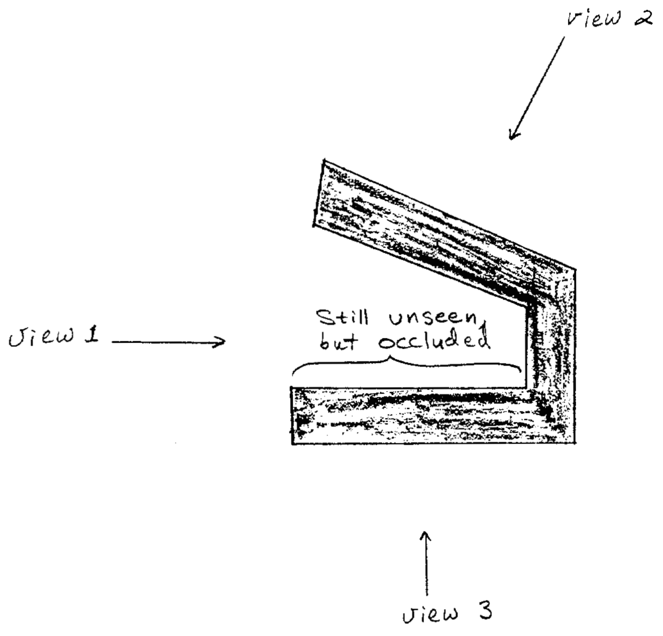

Figure 5. Self-occluding scene