Describing a Robot's Workspace Using a Sequence of Views from a Moving Camera

TSAI-HONG HONG AND MICHAEL O. SHNEIER

***Abstract*—This correspondence describes a method of building and maintaining a spatial respresentation for the workspace of a robot, using a sensor that moves about in the world. From the known camera position at which an image is obtained, and two-dimensional silhouettes of the image, a series of cones is projected to describe the possible positions of the objects in the space. When an object is seen from several viewpoints, the intersections of the cones constrain the position and size of the object. After several views have been processed, the representation of the object begins to resemble its true shape. At all times, the spatial representation contains the best guess at the true situation in the world with uncertainties in position and shape explicitly represented. An octree is used as the data structure for the representation. It not only provides a relatively compact representation, but also allows fast access to information and enables large parts of the workspace to be ignored.**

The purpose of constructing this representation is not so much to recognize objects as to describe the volumes in the workspace that are occupied and those that are empty. This enables trajectory planning to be carried out, and also provides a means of spatially indexing objects without needing to represent the objects at an extremely fine resolution. The spatial representation is one part of a more complex representation of the workspace used by the sensory system of a robot manipulator in understanding its environment.

I. Introduction

A robot moving about in a fixed volume, such as the space above a worktable or conveyor, needs to know where objects are located and which objects occupy each filled volume. To do this, it must build up a description of the space. The proposed method involves using sensed information acquired from arbitrary but known locations to construct the spatial representation incrementally. This is accomplished by projecting the image resulting from each view into the world, and intersecting the views in the following way.

Each object (connected component) in the image projects into the world as a "cone" with its tip at the center of focus of the lens, and its cross section defined by the boundary of the component. The background projects as a cone bounded by the sides of the image with various holes in it for the objects. When two images are acquired from different viewpoints, an object appearing in both images is constrained to lie in the intersection of the cones from each viewpoint. Empty space in either view projects as empty space, while parts of the workspace that have not been seen are explicitly declared to be unknown. The aim is to represent only as much as is known about the workspace at all times. If an object has only been seen once, its position can only be constrained to lie within some cone. If it has been seen many times from different viewpoints, then not only will its position be more tightly constrained, but so will its shape. Eventually, the whole workspace should be represented in a way that closely approximates its true state.

The representation chosen for implementing this scheme is an octree. Unlike earlier work on this problem [2], a single octree is used to represent the whole workspace instead of having a separate representation for each object. The octree is a hierarchical representation based on successive, uniform decompositions of a cube ([4], [6], and [8]). Imagine a cube enclosing the robot's workspace.

Manuscript received October 15, 1984; revised April 5, 1985. Recommended for acceptance by S. L. Tanimoto.

The authors are with the Robot Systems Division, National Bureau of Standards, Gaithersburg, MD 20899.

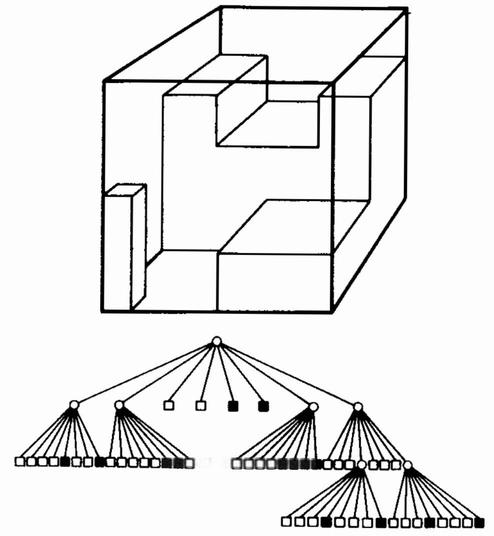

Fig. 1. Objects enclosed within a cube, and the octree representing the volume of the cube.

If the whole volume is uniform (filled or empty), then the single cube describes the volume adequately. If parts of the cube are filled and other parts are empty, then the cube is split into eight octants and each octant is examined for uniformity. The splitting process continues until each (sub-) octant is uniform or until a resolution limit is reached. The set of cubic volumes can be organized into a tree with uniform volumes as leaf nodes, and nodes that must be split as branch nodes (Fig. 1).

II. Alternate Approaches

Approaches to constructing the octree from a sequence of two-dimensional views can be divided into two major categories. The first is characterized by projecting each image into the volume occupied by the tree, while the second involves projecting the nodes in the tree into each image. Both approaches were studied with the conclusion being that it is more efficient to project the octree nodes into the image plane because the cubes always project as convex shapes.

The most obvious way of constructing the octree is to take each component in an image and project it into the tree. On closer analysis, however, it is not obvious how to do this. Consider the general case in which the image plane is arbitrarily oriented with respect to the octree and the cones expand into the octree under perspective projection. There is no direct relationship between regions in the image plane and cubes in the octree. In addition, the resolution of the octree cannot be measured in pixels in the image because the pixel size is a function of the camera's position, while the resolution of the lowest level in the tree is fixed. Ray casting is not feasible because individual pixels also expand into cones, and the same difficulties arise. Attempts at decomposing the image into regions that are meaningful in terms of the octree lead to the approach of projecting nodes from the tree into the image. This is because the shapes that are meaningful in terms of the octree are precisely those shapes that result from projecting the cubes into the image. This approach is discussed in detail below.

An alternative is to project the image into a series of planes parallel to one of the coordinate's axes of the octree, at depths corresponding to the sizes of nodes at various levels in the tree (Fig. 2). This involves deciding which axis is most appropriate for a particular projection plane, and performing a large number of projections. A method that is less complicated makes use of the approximation of the boundaries of components with straight line segments. These segments project as planes into the octree, and convex sets of half-spaces defined by the planes define the cones. Each

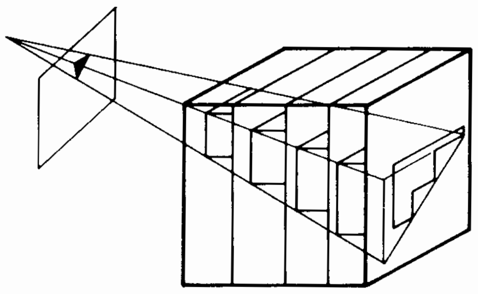
Fig. 2. Projections of an image into planes parallel to one of the coordinate axes of the octree.

cube in the octree is also defined by a set of planes, and the intersections of these planes with each surface of the cones specify whether or not to split a cube into octants, or if it is an object or a background leaf.

Unfortunately, this method is very time consuming, although there are fast algorithms for determining if a surface intersects with a cube [3]. A way of possibly speeding up the process would be to replace the sets of planes with smaller sets of more complex surfaces. The method described by Field and Morgan in [3] allows a decision to be made about the intersection of a second-degree polynomial with a cube. The problem with this approach is that it requires that the image cones be split into subregions that are accurately described by second-degree polynomials (at the very least into convex regions). In the worst case, the method for detecting intersections still involves the solution of a number of equations, and was found to be too slow for practical applications.

Comba [1] gives a more general technique for finding whether or not a set of convex objects bounded by some set of surfaces has a common intersection. His approach is to define a single "pseudocharacteristic function" that is less than zero only in the regions where all the objects intersect. The approach relies on a descent procedure to find the minimum of the function, which is an approximation to the region of intersection. Comba points out that, for the case in which all surfaces are planes, the problem reduces to a linear programming problem. It seems impracticable to have to solve such a problem at each node in the octree for each object cone (although the special orientations of the cube faces might simplify the problem slightly).

The approach finally selected uses the alternative strategy of projecting the cubes from the octree into the image. It allows use to be made of the fact that cubes always project as convex regions, and can be made efficient by using lookup tables. The algorithm is described in the next section.

III. The Algorithm

Assume that the octree has been created from an initial root node. The leaf nodes of the tree can have one of three kinds of labels. A node is labeled "background" if the volume it represents is known to be empty. It is labeled "object *n*" if it may contain object *n*, and it is labeled "unknown" if it has not been seen or was within a cone for an object that is known not to extend into the region. (Note that the label "unknown" is treated exactly the same as that for an object; it is differentiated only to emphasize that the contents of the region have not been seen.) While branch nodes are also restricted to the three kinds of labels, each branch node can store labels for several objects if all the objects intersect with its volume.

The input to the octree intersection algorithm consists of a two-dimensional image, a set of the corner points for each object (component) in the image, and information about the position of the camera in relation to the three-dimensional volume represented by the octree. The two-dimensional image has each connected component labeled with a unique identifier. This makes the intersection tests below particularly easy because it is only necessary to project a point into the image and check the identifier of the corresponding pixel to decide if there is an intersection. The corner information is used to approximate the boundaries of objects for use in defining the cones.

The method involves first deciding which of the cubes in the octree intersect with the cone defined by the boundaries of the image and the center of focus of the camera. This cone extends forward from the camera into the volume represented by the octree. If no cubes (nodes) in the octree intersect with this cone (i.e., if they are behind the camera, or off to one side), then there is nothing to do. Otherwise, some cubes might intersect with the image plane, so that they are partly in front of the camera and partly behind it (if the viewing point is inside the volume represented by the octree). In this case, these cubes are split and the same tests are applied to their children. Those cubes that are within this limiting cone might potentially intersect with cones defined by the objects in the image. Further anlaysis is required in these cases. In essence, this processing requires finding intersections between each cube and the cones for the individual objects (and the background). For efficiency reasons, the intersections are found by first using a number of quick tests and then, if necessary, performing a complete intersection analysis. The procedure involves projecting each cube into the image plane, and all tests are performed in that two-dimensional space.

For each image, the following tests are performed on each octant in the tree.

1) If the octant is behind the image plane, or off to one side, then ignore it.

2) If the octant contains the image plane, then split it into suboctants so as not to process those parts of the octant that are behind the image plane, and perform these tests recursively on each octant.

3) If neither of cases 1) and 2) occur, there may be some intersection between the node and the cones projecting out from the image plane. Instead of projecting the cones out into the tree, however we choose to project each cube into the image plane.

To project a cube, it is necessary to identify the vertices that are visible from the camera's point of view. A lookup table is used for this purpose. The table is constructed by considering a set of volumes surrounding a cube. These volumes are defined by extending the plane faces of the cube, giving rise to a set of half-spaces (Fig. 3). Any viewpoint falls into one of these volumes, which uniquely specifies which vertices are visible. Since the position of each cube is known in world coordinates, and the position of the camera is also known in world coordinates, it is easy to find their relative positions, and so determine from which surrounding volume the cube is being viewed. The vertices are transformed using the world-to-camera transformation which is one of the inputs to the algorithm. This projects the cube into the image plane. Now a set of tests is applied to the projection to decide on a label for the cube. The first tests are quick checks. If they succeed, they save a lot of effort in the interpretation. However, if they fail, a more complete analysis must be performed. The tests, in order of application, are as follows.

a) Check to see if some of the corners of the projection of the cube are inside an object, and some are outside (Fig. 4). If so, store the label attached to the object at the node in the octree corresponding to the cube, and repeat the test on the next object. After testing all the objects and adding the necessary object labels to the node, split the node and apply the algorithm to its children recursively. Note that the only objects in the image that need to be intersected with the projected cube are those that intersected with its parent. (Hence, the need to find all intersections.) At the lowest level in the octree, a special procedure is needed to decide if a node in the tree intersects with an object. This is because the resolution in the tree is lower than that in the image, and it is possible for parts of more than one object to appear in a leaf node. This is described further below.

b) If test a) fails (as in Fig. 5), then check the corners of each object that intersected with its parent against the projection of the cube. If any corner of an object is inside the cube, label the corresponding node in the tree with the name of the object. If the test

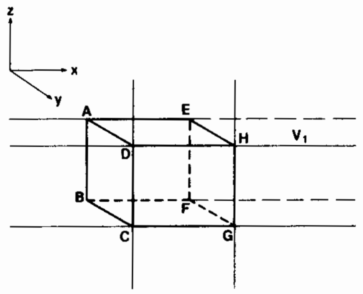

Fig. 3. The six plane faces of a cube divide the surrounding space into 26 half-spaces. Let $V1$ be defined by planes $ADEH$, $BCGF$, $ABEF$, and $DCGH$. Then, from any viewpoint within $V1$, the corners E, F, G, and H are visible.

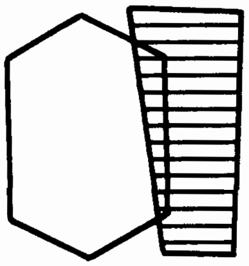

Fig. 4. An example in which some of the corners of the projection of a cube are inside a region, and some are outside.

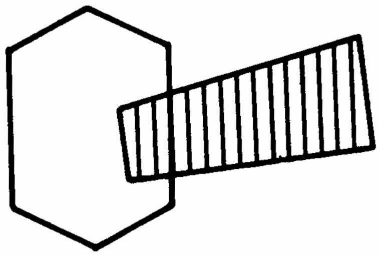

Fig. 5. An example in which some corners of an object lie within the projection of a cube.

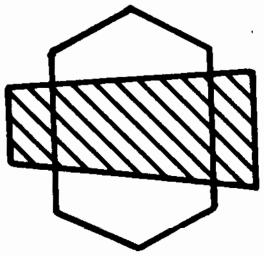

Fig. 6. An example in which no corners of the projected cube lie within an object, and no corners of the object lie within the cube.

succeeds for any of the objects, split the cube and repeat the algorithm on its children.

c) If both tests a) and b) fail (as in Fig. 6), then one of two further tests is performed depending on the size of the cube being projected (i.e., the level in the octree).

i) If the cube is small enough, each boundary point on its projection is checked to see if it is inside any of the objects that intersected with its parent. The checking is done using a binary search, continuing until all points on the edges have been tested or until an edge point is found to intersect an object. In the latter case, the object label is added to the node and the node is split into octants as above. The intersection test is simple. The pixel in the image addressed by the coordinates of an edge point is examined, and if it is an object point there is an intersection otherwise there is not.

ii) If the cube is larger than some threshold, the boundaries of each object are checked for intersection with the boundaries of the projected cube. When an intersection is found, the corresponding label is added to the node and it is split as above. The intersection algorithm is similar to that described in Pavlidis [7], except that some simplifications can be made because only the existence of an intersection need be discovered, not its actual location.

d) If all the above tests fail, then the cube is either entirely wihin an object or entirely outside all objects. These cases can be distinguished using the results of test a) [given that tests b) and c) have already been performed]. If the cube is outside all objects, it is labeled as a background leaf node. Otherwise, the node is a leaf node for the object in which it lies.

After the first images have been projected, branch nodes may have several labels while leaf nodes still have only one. As new image cones are projected, the information must be updated. Two procedures are followed for updating the labels at a node, one for leaf nodes and one for branch nodes.

For leaf nodes, the labeling is straightforward (see Table I). If the current view (the set of cones arising from the current image) does not include a node, the node retains whatever label it had before. If the current view interprets a node as background, the node is labeled background. If the label at the current node agrees with the label assigned from the current view, there is again nothing to do. Otherwise, there is a conflict. If the leaf node is at the highest resolution, the node can be labeled as one or the other object according to what percentage of the node is filled by the projection from the current view, or depending on the color of the projection of the center of the node. For larger leaf nodes, both labels can be retained with confidences related to how often they have been confirmed, or adjacent leaf nodes can be examined to establish a majority opinion.

Branch nodes are somewhat more complicated because of the possibility of affecting whole subtrees when making a decision about a node (see Table II). If the new information confirms that the node is a branch node, all that needs to be done is to add the labels of any new objects found within the volume to the set of labels at the node. If the new information interprets a node as a leaf node, then the action to be taken depends on what kind of leaf node is hypothesized. If the new label is background, then the branch node is converted to a background leaf node and all its children are erased. If the new information says that the node should be an object leaf node, some further interpretation is necessary. If the object label is not among those already known to be within the corresponding volume, it must be added to that list. The extra processing described below arises because of the nature of the intersection process. It is performed if the new information interprets a node as an unknown or an object.

Every branch node contains a list of all objects in its volume. From two views, an object is constrained to lie in the intersection of the two resulting cones (and to regions visible in one view but not in the other, but this situation is handled automatically by the algorithm). To be safe, regions in cones that are labeled with the object name but do not appear in the intersection, are labeled unknown. This takes care of the possibility that another object is in front of, or behind the object, and totally contained in its projected cone. If the old information in the tree says that the object appears in a volume, but the new information does not, then the leaf nodes in that volume are labeled unknown, while branch nodes have the object's name deleted from their list of labels.

All the processing of the image and the updating of the tree can be done in a single pass, with new labels replacing old where appropriate and all cones being simultaneously projected. At every branch node, a check is made to see if all the children are leaf nodes with the same label. Such nodes are merged into their parent which becomes a new leaf node.

IV. Discussion

The problem of constructing a description of an object from a number of views has a long history. Almost all previous work has been concerned with constructing descriptions of single objects,

TABLE I
THE LABELING PROCEDURE FOR LEAF NODES IN THE TREE

Label from Current View	Previous Label	New Label
Unknown	(anything)	Previous Label
Background	(anything)	Background
(anything)	Background	Background
(anything)	Unknown	Current Label
Object *X*	Object *X*	Unchanged
Object *X*	Object *Y*	Depends on the Relative Confidences

TABLE II
THE LABELING PROCEDURE FOR BRANCH NODES IN THE TREE

Node in Old Tree	Node in Current View	New Node
Branch	Branch	Branch
Background Leaf	Branch	Background Leaf
Other Leaf	Branch	Branch
Branch	Background Leaf	Background Leaf
Branch	Other Leaf	Branch

however, and not with describing a whole scene. There are a number of simplifications possible when describing objects that are not applicable to the case of describing a scene. For single objects, a coordinate system can be defined that is appropriate for the object, and the data representation can be tailored to the object. The problems of occlusion that arise are only those of self occlusion which has not usually been considered. There are no possibilities of touching objects, nor of ambiguities of interpretation.

Early work was performed by Underwood and Coates [9]. They constructed descriptions of convex polyhedral objects in terms of surfaces, edges, vertices, and connectivity. An object was placed on a turntable and a succession of images was acquired, with the requirement that there be some overlap in the surfaces visible in successive views. The common surfaces were then matched and the new surfaces added to the graph describing the objects The resulting description accurately represented the shapes of the objects that were scanned.

More recently, Martin and Aggarwal [5] describe an algorithm to construct a description of an object from multiple two-dimensional views by projecting cones into an object-centered coordinate space. They used parallel projection and constructed a representation, called a volume segment representation, by projecting the cones arising from each view into each of the coordinate axes in the object space. This technique could be applied to the more general problem of describing a scene, but the representation would probably require much more space. It is similar to the approach of projecting each image into a series of planes parallel with the octree coordinate system and then constructing the intersections.

Connolly [2] describes a system that uses a set of range data images to construct an octree describing an object. He constructs a quadtree description from each image and projects the quadtree blocks, using parallel projection, into the octree. He makes the assumption that quadtree nodes are directly comparable to octree nodes, implying a fixed distance between the camera and the octree origin, and constructs labels for octree nodes in a manner very similar to that described in this correspondence. His procedure appears much slower than that described here (ten minutes per range image), however, and is less general than our method.

A number of problems can arise due to difficulties in accurately positioning and calibrating the camera, and because of noise in the images. The current calibration reduces the measured positioning errors of objects to about 2 mm at a range of 1 m (using a technique of matching object features to models to compute the object's position in camera coordinates, and then transforming from camera to world coordinates). This is due to the angle subtended by one pixel at that range, and is adequate for the relatively coarse spatial representation for which the octree is constructed. Periodic recalibration of the camera and robot arm are necessary to ensure that errors are kept small. This can be done on the fly by observing fixed calibration points in the world.

The problem of errors due to noise in the image is perhaps more serious. Noise that makes object appear larger, or introduces spurious objects is not a problem because subsequent views should not produce the same noise patterns. Noise that increases the size of the background, however, is more of a problem because regions labeled background remain background in subsequent views, even if later views show them to be filled. Projecting recognized objects into the tree from their geometric descriptions can alleviate this problem, as can algorithms that deal with moving objects in the tree. Currently, however, we are considering a scheme where evidence about the contents of a node can accumulate and eventually change the label at the node after a sufficient number of views indicate that the label is incorrect. This can be implemented with no basic changes in the algorithm described above.

The main advantages of using an octree for representing spatial information in a form suitable for navigation, and for spatially indexing objects, are its relatively compact representation and the ability to rapidly index and modify any part of the tree. Trajectories can be directly represented in the tree and, in the process of their construction, collisions can be easily discovered. For controlling a robot, it is also useful to feed small portions of the path to the controller, and this can be done simply by stepping through the nodes in the tree in the correct order. The sizes of leaf nodes also give an indication of the amount of error allowed in the trajectory before any collisions occur.

The algorithm described above is still not as efficient as it could be. A major reason is that each cube is projected independently into the image. This means that each vertex may be projected up to eight times, and the intersection test performed eight times on the same point (eight cubes meet at each vertex). By remembering which vertices have been projected, and what the results were of their intersections, significant savings could be obtained (at the cost of some extra bookkeeping). Another source of inefficiency results from not taking advantage of occlusion information. A decision made about a node in the tree that occludes other nodes from a particular viewpoint is also valid for the occluded nodes. Thus, the projection of those nodes need not be performed although they must still be updated.

V. EXAMPLE

For the example, three pictures of a rectangular block were taken from three different positions. The position of the camera was not known exactly but was computed using a method based on fitting features from each image to a model of the object and then computing the object-to-camera transformation. There is thus some uncertainty about the exact location of the object and the computed position of the camera for each image.

Figs. 7, 8, and 9 show the three views of the object, and the three cones arising from projecting these images into the octree. Fig. 10 shows the result of intersecting cones from the first two images, giving rise to a new cone in the octree that constrains the location and shape of the object. Fig. 11 shows the result after intersecting the cone in Fig. 9 with that in Fig. 10. It can be seen that the shape of the cone in the tree is converging on the shape of the three-dimensional object that is being imaged. In this example, the process could clearly be continued until the exact shape and position of the object were recovered (Fig. 12). In general, this is not true, either because of occlusion problems or because of concavities in the objects. Some of these problems can be overcome by viewing the objects from special positions, or by using a range sensor, but there exist objects that cannot be completely and accurately described using this technique. For the purposes for which this algorithm was developed this is not a serious problem, both because complete descriptions of recognized objects are available and can

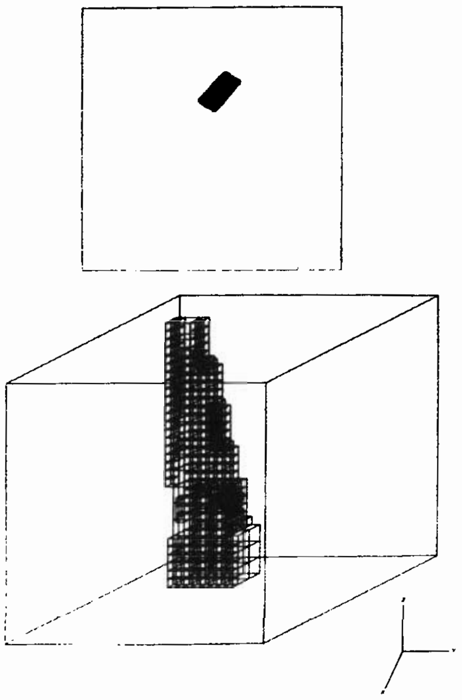
Fig. 7. *Top:* The two-dimensional image of a rectangular parallelepiped. The view is almost straight down. *Bottom:* The cone projected into the octree from this image.

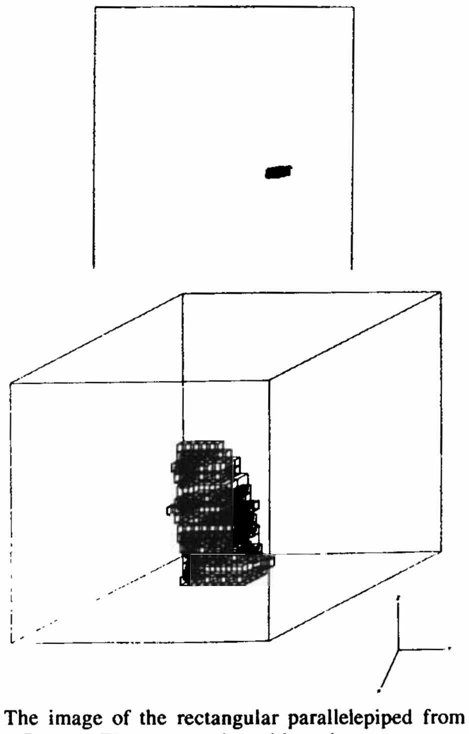
Fig. 8. *Top:* The image of the rectangular parallelepiped from one side. *Bottom:* The cone projected into the octree.

be projected into the octree and because the representation is not used for describing fine details of the shapes of objects. Note that there is no requirement that the objects in the world have surfaces parallel to the coordinate axes of the cube. This simply makes the example pictures easier to understand.

Projecting and intersecting a view resulting in 1500-2000 object leaf nodes currently takes under 3 min on a VAX 11/780. Custom hardware being constructed to perform the matrix manipulations for the projections is expected to very substantially reduce this time. It is also expected that simply using a dedicated microprocessor and a large amount of memory to store the tree would have a significant effect. This result compares favorably to that of Connolly [2], especially since we are solving a more general problem than he is.

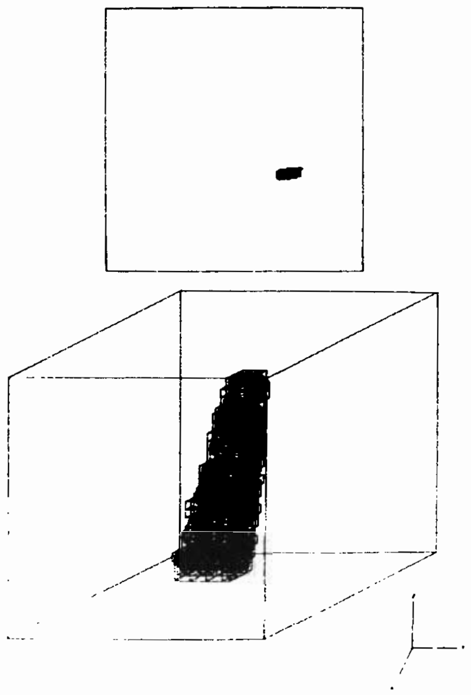
Fig. 9. *Top:* A third view of the object. *Bottom:* The cone projected into the octree.

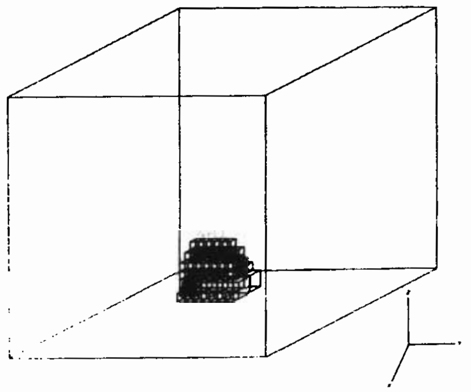
Fig. 10. The result of intersecting the cones arising from the images in Figs. 7 and 8.

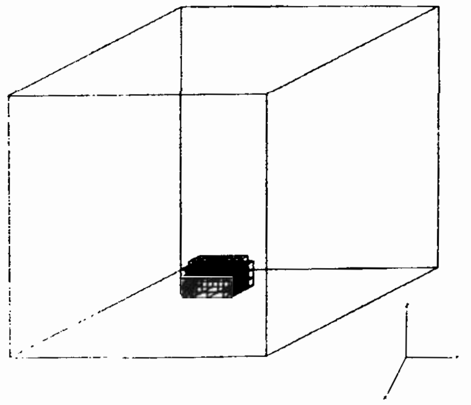
Fig. 11. The result after intersecting the cone in Fig. 9 with that in Fig. 10.

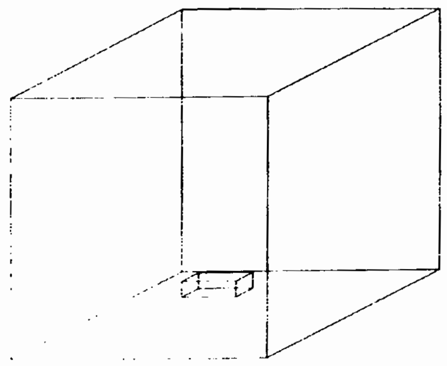

Fig. 12. The ideal result of projecting a large number of views into the octree.

VI. Conclusions

This correspondence has described a means of constructing a spatial representation of the workspace of a robot. An octree is used as the data structure for the representation. It provides a relatively compact representation, allows fast access to information, and enables large homogeneous parts of the workspace to be ignored. The representation is constructed from successive images taken by a camera mounted on the wrist of a manipulator. The manipulator moves about while performing its task, giving rise to a sequence of images taken from different, but known, positions. A series of cones, whose apexes are at the center of focus of the camera lens, and whose cross sections are defined by the boundaries of the objects in the image, are projected into the octree. They describe the possible positions of the objects in the space and delineate the regions that are empty. When an object is seen from several viewpoints, the intersections of the cones constrain the position and size of the object. After several views have been processed, the object begins to resemble its true shape. At all times, the spatial representation contains the best guess at the true situation in the world, with uncertainties in position and shape explicitly represented.

The purpose of constructing this representation is not so much to recognize objects as to describe the volumes in the workspace that are occupied and those that are empty. This enables trajectory planning to be carried out, and also provides a means of spatially indexing objects without needing to represent the objects at an extremely fine resolution.

The algorithm described in this correspondence differs from previous approaches in that it constructs a single octree to describe all the objects in the workspace. It can accommodate views taken from arbitrary viewpoints, and it maintains the octree at a fixed resolution that is independent of the resolution of the two-dimensional images used in its construction. The construction algorithm is also new, and operates by projecting the cubes of the octree into each image, instead of projecting the images into the tree. This was found to be more efficient, largely because the nodes of the octree always project as convex objects and their shapes are known in advance.

References

[1] P. G. Comba, "A procedure for detecting intersections of three-dimensional objects," *J. Ass. Comput Mach.*, vol. 15, no. 3, pp. 354-366, 1968.

[2] C. I. Connolly, "Cumulative generation of octree models from range data," in *Proc. Int. Conf. Robot.*, Atlanta, GA, Mar. 1984, pp. 25-32.

[3] D. A. Field and A. P. Morgan, "A quick method for determining whether a second-degree polynomial has solutions in a given box," *IEEE Comput. Graph. Appli.*, vol. 2, pp. 65-68, May 1982. (Also, General Motors Res. Lab. Rep. GMR-3656, Apr. 1981.)

[4] C. L. Jackins and S. L. Tanimoto, "Octrees and their use in representing 3D objects," *Comput. Graph. Image Processing*, vol. 14, pp. 249-270, 1980.

[5] W. N. Martin and J. K. Aggarwal, "Volumetric descriptions of objects from multiple views," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-5, pp. 150-158, Mar. 1983.

[6] D. Meagher, "Octree encoding: a new technique for the representation, manipulation and display of arbitrary 3D objects by computer," Dep. Elec. Syst. Eng., Rensselaer Polytech. Inst., Troy, NY, Tech Rep. TR-IPL-111, 1980.

[7] T. Pavlidis, *Algorithms for Graphics and Image Processing.* Rockville, MD: Computer Science Press, 1982, Ch. 15.

[8] S. N. Srihari, "Representation of three-dimensional digital images," *ACM Comput. Surveys*, vol. 13, no. 4, pp. 399-424, Dec. 1981.

[9] S. A. Underwood and C. L. Coates, "Visual learning and recognition by computer," Inform. Syst. Res. Lab., Univ. Texas, Austin, Tech. Rep. TR123, 1972.