# A COLLISION DETECTION ALGORITHM for TELEROBOTIC ARMS

Doan Minh Tran
ST Systems Corporation
4400 Forbes Blvd
Lanham, MD 20706

Maureen O'Brien Bartholomew
Code 735
NASA Goddard Space Flight Center
Greenbelt, MD 20771

**Abstract**

In this paper, the telerobotic manipulator's *collision detection* algorithm is described. Its applied structural model of the world environment and template representation of objects is evaluated. Functional issues that are required for the manipulator to operate in a more complex and realistic environment are discussed.

## 1 Introduction

*Collision detection* is the process of detecting imminent collision between moving objects with one another, or a moving object with stationary objects. In a telerobotic environment, detection is concerned with not only collisions between a robot and its surrounding objects but also collisions with itself (i.e., collision between arm's links). Moreover, it involves distinguishing between *unintentional collisions* and *intentional contact* with objects in space. Consider the fact that an end-effector contacting an object such as a coffee mug would not constitute a collision if its goal is to pick up the mug. On the other hand, a collision would occur if the end-effector (or another part of the arm) was to hit any other close-by objects, such as a book located next to the coffee mug. Thus, the collision detection problem requires robot environment awareness on one hand; while on the other, it demands detailed knowledge of objects' characteristics to avoid collisions or to make contact. In other words, collision detection is a process of differentiating between collisions and contacts.

Currently, our collision detection problem is defined by a telerobotic system at NASA Goddard Space Flight Center (GSFC) which consists of two slave Robotics Research Corporation (RRC) arms. Its implementation is part of the safety system which is proposed to serve independently as a redundant monitor of the control system. In addition to redundant collision checks, the safety system performs redundant monitoring of the safety parameters such as velocities, accelerations, torques, and motor currents of the RRC arms. Based on these parameters, it should safely shut down the robot if an attempt is made by the operator to exceed their allowable limits. It also has the capability to override the automatic telerobotic safing functions from the workstation or from other systems. The RRC arm is a seven degree of freedom manipulator, with cylinder shaped links as shown in Figure 1. Movement of the links are rotational in joint-space coordinates. Presently, the arms are in an environment which can be roughly represented as cylindrical or rectangular-shaped objects. In particular, the environment is sparse and well defined with various stands supporting either cameras or Orbital Replacement Units (ORUs). The tasks consists of using the RRC arms to picking, moving, and placing ORU boxes from one location to another. Figure 2 illustrates a typical environment in which the arms operate. It should be noted that the environment is dynamic as the result of ORU boxes being moved by the arms. Therefore, updating of objects' location is necessary.

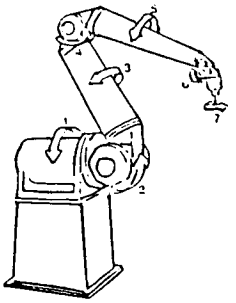


Figure 1: An RRC arm. (Robotics Research Corp. Milford, OH 45150)

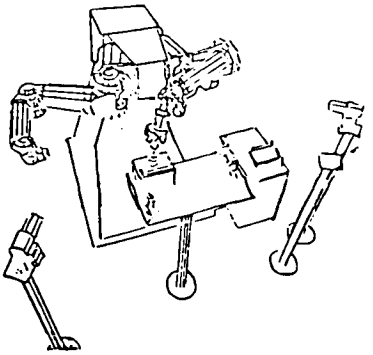


Figure 2: An example of the environment in which the RRC arms operate

## 2 Collision Detection Algorithm

Having defined the collision problem that may occur in a telerobotic environment and identified the contraints of the manipulators, we will now discuss the collision detection algorithm. In this section, we describe the octree structure for modelling the world and detecting imminent collisions followed by a discussion of an object template representation for distinguishing between intentional contact and unintentional collisions.

### 2.1 The Octree

In the following paragraph, we describe the octree structure, decomposition of the robot's workspace into regional octree nodes, functional updates for detecting imminent collisions, and node adjustment to reflect environmental changes.

An *octree* is a data structure encoding a space as a tree of either empty nodes or one which consists of a root node and eight disjoint nodes, each of which can be another octree.

The definition of an octree that has just been given illustrates three points. First, an octree of empty nodes is used to indicate object-free space. Second, a node is decomposed into eight sub-nodes (called *octants*) when its contents satisfies some predefined criteria for refining the resolution (this is referred to as the *decomposition rule* which is discussed later). Third, it exhibits recursive inheritance, that is, each octree node is a sub-octree.

Since we are dealing with spatial index octrees, it is convenient to introduce the distinction between them and image representation octrees. In image processing, octree representations are used to define the shape of object by decomposing and representing object's vertices, edges, and planar surfaces as nodes (this is known as a *polytree* [2]). Another way of describing object features is by subdividing the volumes until all leaf nodes are either empty or fully occupied by that object's bounding surface (this is referred to as a *region octree* [7,6,3]). In this sense, the octree node is used to denote an object's shape, as well as for storing object properties such as color and density. A spatial indexing octree, on the other hand, is used to encode the space (in this case, it is the manipulators' workspace). The spatial indexing octree divides the workspace into a set of cubes

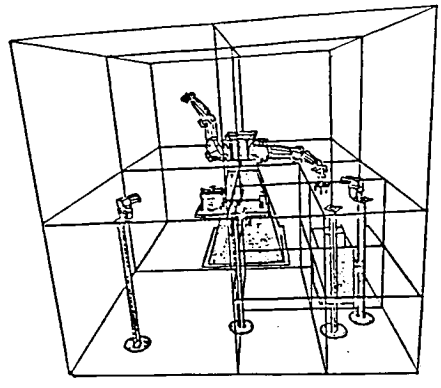
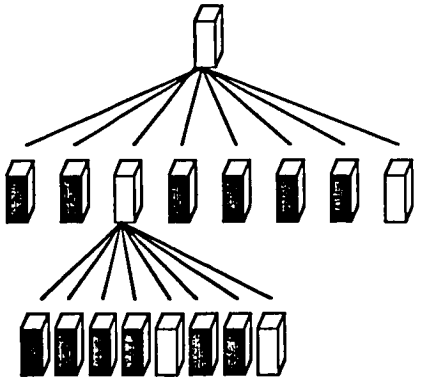


Figure 3: The workspace is partitioned into cubes of various size.

in various volumes. For the sake of consistency, we will refer to the cubes as nodes. Each node, in turn, could either be empty or contain one or more objects. In this context, nodes in an octree denote volumes while their contents hold a list of objects within that regional space [4].

Figure 4 explains our use of the octree. The robot's workspace which consist of a manipulator and various objects can be enclosed in an imaginary cube. This cube is subdivided into eight equal regions, and numbered as shown in Figure 3. In the octree representation, the whole workspace would be denoted as a root node with each sub-region as a sub-node. The workspace in each sub-region can again be subdivided as before and associated with sub-nodes. This process can be repeated until all leaf nodes are either empty or contain no more than three object within it (assuming that the resolution criteria used here allows less than four objects per node). The final octree is shown in Figure 4.

In brief, image representation octrees require a separate octree representation for each object while spatial modeling needs a single octree for encoding all objects. Consequently, spatial index octrees require different methods for splitting nodes than image representation octrees. In the following paragraphs, two methods of decomposing spatial indexing octrees, the *compat-*



□ = Empty Space
■ = The region is filled with objects

Figure 4: The Octree representation of the correspondence workspace.

*ibility decomposition rule* and *n-objects rule*, are introduced.

In order to grasp the concept of the *compatibility decomposition rule*, we need to understand the notion of *compatible* objects. According to Schaffer and Herb [4], objects or parts of objects (the term *primitive* is used in their paper to covey both object and parts of an object) are compatible if it is impossible for them to collide with each other. For example, an ORU box sitting on top of a stand could not be considered as a collision between the ORU and the stand; thus, these objects are compatible. Mutual compatibleness also exists between any two geometrical abutting links of the manipulator, assuming that the servo level controllers do not allow an angle less than that which would cause the two links to come into contact. In brief, the compatibility decompostion rule dictates the subdivision of an octree node into sub-nodes only if it contains objects that are not mutually compatible. In addition, the compatibility rule appears to involve less octree-updating in a static environment, because updating is only required when a new object is introduced into any region. It is not obvious, however, how compatibility among objects is determined in a dynamic situation. Consider our previous example, compatibility exists between the ORU on top of the stand; but what about picking the ORU box up and then dropping it onto the stand. Clearly, this free-falling/contact action would be considered as a collision between these two objects. Thus, the test of compatibility among objects involves both functional knowledge of objects as well as knowledge of the task being performed.

An alternative decomposition rule might be to subdivide a node if it contains more than "**n**" objects. This rule is similar to the region octree that has been discussed before. Instead of subdiving a node when an object's volume partially fills that region, nodes are split until all leaf nodes containt less than or equal to **n** objects. From the outset, this rule tends to require a lot of updating of the octree as objects are moved from one place to another, regardless of whether they are moved inside or outside of their current regional node. In light of this, the *n-object* decomposition rule is faster than the *compatibility* rule, since it involves no knowledge processing and updating is done only to nodes that contain the moving object(s).

As mentioned before, octree updating is performed everytime objects (i.e., the robot arm, part of the arm, or a box) are displaced. In this context, updating involves both modifying the content of those nodes and checking for collision among

objects within a regional node or with neighboring regions.

When objects are moved into another region, node content must be updated. This involves removing the object from its current node and inserting it into the new region. To locate the nearby node, the *neighbor-finding* technique (a traversing technique for locating object's neighboring regions in octree [8]) is used. Neighbor-finding works by first locating the octree node that contains the desired object; then, begin traversing up the tree until a nearest common-ancestor for both the object's node and its desired neighbor node, is found. From that common-ancestor, it descends in a mirror-like direction while ascending the tree. The final stop will be the node that represents the object's neighboring region. It should be noted that, node insertion might change the octree structure. This depends on the decomposition rule that one uses in the algorithm.

According to Boyse [1], detecting a collision for a pair of objects can be done by interference checking of an object's edge with a face of the other or vice-versa. In doing so, one of two things may occur: an object's edge passes through the interior of the other object's face; or it contacts the other object's face boundary. For the former, collision can be detected by determining the locus of each endpoint of the moving edge and examining these loci (space curves) to see whether any intersect the face. In the later case, collision is detected by examining the boundary of the face to see if it intersects the surface generated by the moving edge.

In summary, spatial indexing octrees are useful for detecting imminent collision within the robot's workspace, by encoding the environment as a set of nodes (i.e., cubes) with various volumes. Each node could be subdivided into a set of octants for better resolution (or for manipulation), if its contents satisfy a decomposing criteria. When objects are moved (unless the compatibility decomposition rule is applied and the objects are moved within its current region), the edge-face algorithm is applied to check for collision among objects, and the nodes' structure and/or content are modified to reflect the changes in the environment.

## 2.2 Template Representation

The problem of distinguishing intentional contact from unintentional collision of objects can be resolved by relying on the system's knowledge of the objects' role with respect to the task. In other words, objects can be categorized from the task as: (1) objects that are manipulated by the telerobot's manipulator and thus come in to contact with it; (2) objects that are caused to collide intentionaly with other objects by the manipulator; or (3) objects which are neither manipulated by the end-effector, nor collided with any other objects. All other types of contacts are interpreted as unintentional collisions.

Imagine an ORU box laying upright on a table. Suppose that in addition to the ORU's position and size, the system also knows it to be a manipulative object (i.e., being picked-up, moved, or placed at other locations by the telerobotic manipulator). The table is viewed as a contacted object. Now, as the end-effector approachs the predefined collision range of the ORU, the collision detection system would assume that the telerobotic's goal is to pick-up that object; thus, it does not view this as an unintented collision (and prohibit any further advancement of the arm). Rather, it allows the arm to proceed at a slower velocity and eventually empowers the end-effector to come into contact with the ORU. The same procedure could be applied to the situation where the manipulator places the ORU on the table; it would not view the contact between these two objects (the ORU and the table) as unintentional collision. However, contact between the telerobotic manipulator and the table *would* be seen as an unintentional collision since the table is viewed by the robot as unmanipulative. Thus, system knowledge of objects' roles enables it to differentiate between intentional contact and unintentional collision. Let us discuss how knowledge of objects' characteristics could be represented internally.

Objects can be defined in terms of their primitive role as: *supporting*, or *manipulating*, or neither; in addition to their shape, size, and position. The *supporting* role denotes an object that can be collided with by another object, providing it is stationary before and during the time of collision. For example, the table illustrated above is defined as having the *supporting* role, and the ORU as having the *manipulating* role. While the *supporting* role allows collision between two objects other than the telerobotic manipulator, the *manipulating* role permits an object (or area of an object) that the end-effector of the robot arm can collide with. One can specify whole or part of object as supportive or manipulative.

More formally, objects can be defined as follows:

```
(object-name
  (primitive-role dimension position))  or
(object-name
  ((component-name
      (primitive-role dimension position))
   (component-name
      (primitive-role dimension position))
   (      :      ))
   dimension position))
```

For example, take a (30"Wx42"Lx38"H) table that is located 60"x50" away from the arm. Its (30"x42"x4") table-top has the *supporting* role (it allows other objects to contact). The remaining components of the table must be protected from contact. It could be described as:

```
(table
  ((table-top
     (supporting (30 42 4) (60 50 34))
   (30 42 38) (60 50 0))
```

Given the object representation above, we intend to solve the problem of differentiating intentional and unintentional collision. Consider, for example, where the manipulator approaches an ORU that is located on a table. Under the current situation, the system predicts an imminent collision: between the arm and the table; and between the end-effector with the ORU. A search of object characteristics in the database indicates that a *supporting* role was assigned to the table, while the ORU object has a *manipulating* role. Based on this information, the system assumes the user intends for the end-effector to contact the ORU but not the table. Thus, it places certain constraints on future movement of the arm. One possible constraint would be for the system to decrease the arm's velocity toward the ORU box; and to inhibit further advancement in the direction of the table.

Another problem in differentiating unintentional collisions from intentional contacts is where the arm holding an object (such as, a ORU box) is about to collide with a stationary object. In the case where the stationary object is supportive (i.e., a table), then it is solvable by assuming that the telerobot's intention is to place the holding object on it. But if the stationary object is manipulative (like another ORU box); then, the event would be declared as an unintentional collision.

Another advantage of this object template representation is, it allows us to logically manipulate parts of an object as unique entities, while it retains the physically inseperable aspects of these components as they make up an object. Hence when an object is moved, all of its components are moved.

In short, knowledge of objects' characteristic (such as *supporting* or *manipulating*) enables the system to predict human operator's intention for the manipulator. Thus, it can inhibit or permit further advances of the arm. Such an approach, however, might lead to collisions of objects due to incorrect assumptions. Nevertheless, these collisions would most likely cause only small physical damage, since the velocity of moving objects are forced by the control system to be very small. A more fail-safe approach to recognizing intentional contact from unintentional collision is to querry the human operator, at the first sight of an imminent collision. However, this would require tedious interaction between the system and the operator, slowing down task performance.

## 3 Conclusion

In conclusion, a collision detection algorithm for a telerobotic environment must have the ability not only to detect imminent collisions, but also the capability to differentiate between an intentional contact and an unintentional collision. In this paper, we have introduced an octree structure approach to detect imminent collisions. It is a divide-and-conquer algorithm that decomposes the robot workspace into sub-regions. Each sub-region can be empty or occupied with objects. When an object is moved, its edges' intersection with other objects' faces (or vice versa) are calculated in order to detect an imminent collision in the region. On the whole, the spatial index octree approach provides a relatively structured and compact representation, allows a large portion of the workspace to be ignored, and enables real-time updating [5]. However, these advantages depend on the decomposition rules, and those in turn are dictated by the environment and the tasks to be performed.

Once an imminent collision between objects is detected, the system must decide what action should be taken: either stop the telerobot manipulator from further advancement, or set some constraints on the movement of the arm. In order to make this decision, the system must recognize intentional con-

tact and unintentional collision. One solution to this problem is by relying on knowledge of the objects' role with respect to the telerobotic task. For our particular task, objects are defined by their primitive role of either *supporting* or *manipulating*; in addition to their shape, size, and position. Based on this knowledge, the system could infer certain assumptions about the telerobot's intentions as it approaches objects. Consequently, it signals the control system to decreases the arm's movements (in the case of intentional contacts) and outputs warning message, or it inhibits any further advanced of the robot (if it is unintentional collisions). System knowledge of these objects' characteristics help reduce tedious interaction required of the users. However, there is a cost to such approaches. It might lead to collisions of objects due to incorrect assumptions by the system.

## 4 Discussion

The issue of distinguishing unintentional collisions from intentional contacts has been addressed. What has not been addressed is the issue of interfacing between the collision detection algorithm and other sub-systems within the telerobotic system. In particular, issues that involve describing the world model, database of manipulated and displaced objects by the telerobotic's arm, and handling collisions between objects. Thes problems need to be resolved in order for the manipulator to operate safely and efficiently in a more complex and realistic environment. In the following paragraphs, we address these issues in hope that further research will be conducted to shed some understanding on the problems and their solutions.

The issue of efficient versus effective safeguarding of the operation of the telerobotic manipulator lies partly on the representation of object. In generalizing objects as either solid rectangulars or solid cylinders, we can in effect increase the performance of the safety system due to the simplification of computing objects. By doing so, on the other hand, we have constrained the arm to operate effectively on objects. View a table as a solid rectangular object, for example, we would reduce the computational time describing and detecting collision of objects. But we also inhibit the arm from operating in the space which is under the table. Another classical problem is, how to describe contained objects, such as, a camera in an ORU box. In addition to the redundancy of computing objects, if we differentiate them, we are faced with the problem of processing knowledge that the displacement of the camera might not alter the position of the box but not vice versa. However, if we initialy define the ORU box and its contained camera as one whole object; then any attempt by the telerobotic arm to access the camera will not be possible, since it is viewed as a collision of the arm with the ORU.

Another problem involving safety issues is, when an object is manipulated and displaced by the end-effector. Under this scenario, dynamic or real-time updating of that object's orientation and position are required to detect any imminent collision between the object and other stationary objects or with the arm. It also demands knowledge of which events would cause alternation in object shape while its orientation and position remain fixed. Take an ORU box, for example, where its door is opened by the end-effector. This event requires detection of any collision between the open door with objects (including the arm) that are in its path. It also requires system knowledge that updating the object should only be focused on its shape and not on its position or orientation. In contrast, only the orientation and position of the ORU box are required to be modified, if the arm moves it to another place.

One final issue is one of handling collisions. This problem involves not only when and how to stop moving objects (particularly the telerobotic arm); but also the issue of what information regarding the current environment should be retained prior to system (or components) shut-down must be considered as part of handling a collision. This is due to the fact that, system restarting requires information of the current world.

## Acknowledgements

We would like to thank Ravi Kaipa for editing this paper. Thanks are also due to, in alphabetical order: Russ Byrne, Christ Shenton, Scott Swetz, and Janice Tarrant for their helpful comments and suggestions. Research sponsored by NASA Goddard Space Flight Center, Greenbelt, MD.

## References


[1] Boyse J.W., *"Interference Detection Among Solids and Surfaces,"* Comm. ACM, vol 22, no. 1, Jan 1979, pp. 3-9.

[2] Carlbom I., Chakravarty I., and Vanderschel D., *"A Hierarchical Data Structure for Representing the Spatial Decomposition of 3-D Objects,"* IEEE Comp. Graph. Applications, vol 5, no. 4, Apr 1985, pp. 24-31.

[3] Glassner A.S., *"Space Subdivision for Fast Ray Tracing,"* IEEE Comp. Graph. Application, vol 4, Oct 1984, pp. 15-22.

[4] Herb G.M. and Shaffer C.A., *"A Real Time Robot Collision Avoidance Safety System,"*

[5] Hong T. and Shneier M.O., *"Describing a Robot's Workspace Using a Sequence of Views from a Moving Camera,"* IEEE Trans. Pattern Anal. Machine Intell., vol PAMI-7, no. 6, Nov 1985, pp. 721-726.

[6] Jackins C.L. and Tanimoto S.L., *"Oct-Trees and Their Use in Representing Three-Dimensional Objects,"* Comp. Graph. Image Processing, vol 14, no. 3, Nov 1980, pp. 249-270.

[7] Meagher D., *"Geometric Modeling Using Octree Encoding,"* Comp. Graph. Image Processing, vol 19, no. 2, Jun 1982, pp. 129-147.

[8] Samet H., *"Neighbor Finding in Images Represented by Octrees,"* Comp. Vision, Graphics, and Image Processing, vol 46, 1989, pp. 367-386.