Coarse-to-Fine Collision Detection for Real-Time Applications in Virtual Workspace

Yoshifumi KITAMURA†, Haruo TAKEMURA†† and Fumio KISHINO†

† ATR Communication Systems Research Laboratories
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-02, Japan
phone: +81-7749-5-1211
<kitamura, kishino>@atr-sw.atr.co.jp

†† Nara Institute of Science and Technology
8916-5 Takayama-cho, Ikoma-shi, Nara, 630-01, Japan
phone: +81-7437-2-5291
takemura@is.aist-nara.ac.jp

Abstract — We propose a method for detecting interference and potential collisions among objects to facilitate cooperative work in a virtual space. It satisfies both "accuracy" and "computational efficiency" in the virtual environment where there are multiple independently moving objects. In paticular, the method detects faces with arbitrary motion (translation and rotation) in three main stages. In the first stage, the coarse stage, an approximate test is performed in the entire workspace to select sufficiently close objects by using their bounding shapes. In the second, the mid stage, interfering object parts are identified using the octree representation of the object shapes. In the third, the fine stage, a polyhedral representation of the object shapes is used to more accurately identify the faces that are likely to collide. Specific pairs of faces belonging to any of the interfering objects found in the second stage are tested; detailed computation is therefore performed on a reduced amount of data. This method tests collisions in the entire workspace all the time, and achieves an appropriate efficiency in detecting pairs of faces about to collide. Experimental data demonstrate the efficiency of the proposed method.

Keywords — virtual workspace, operator assistance, collision/interference detection, bounding shapes, octree, polyhedral shape representation

1 Introduction

A virtual environment created by computer graphics and having appropriate user interfaces can be used in many applications. For example, by measuring a user's viewing position, hand position and hand shape, we can establish an object manipulation environment in a virtual space where the user can directly grasp, move or release objects created by stereoscopic computer graphic images. Provided multiple users could share this environment, more potential applications would exist [1]. In such a case, it would be ideal for the users to handle objects just as they would do in a real environment with no sense of incompatibility.

When two or more operators are engaged in cooperative work such as changing a layout, matching parts, etc., if their work is supported by assistance tools, they may be able to manipulate virtual objects without difficulty. Such tools include a force feedback device, which can generate a reaction between two faces touching each other [2], and guidance software which can lead faces of objects to appropriate positions [3]. In any case, it is important to select the attracting pair of faces of objects and test the interference or collisions among them efficiently.

The issue of interference or collision detection has been examined in studies focusing on path planning of manipulators or mobile robots [4][5][6][7]; however, a drastic increase in computation results when detection techniques are applied directly to the problem in a virtual workspace. When we consider the application of a collision detection method to real-time operator assistance in a virtual workspace, investigation into the computation time is necessary so that the operator will not feel any sense of incompatibility. For the applications of computer animation and virtual reality, much research effort has been directed to efficient interference or collision detection [8][9][10][11]; however, they have not been able to efficiently identify the attracting pair of faces that are likely to collide in the entire workspace containing multiple moving complicated objects.

Collision detection methods using polyhedral shape representation are common and yield comparatively accurate results, but they need to test all combinations of faces and edges of objects in the environment to detect interference. Therefore, the computational cost increases with the number and shape complexity of objects. Though [12] uses vox-

els for rough identification of the candidat faces, spatial access is not efficient; this is because it uses a flat set of voxels (i.e., all voxels are the same size) rather than a hierarchical structure (such as an octree), which could be used to quickly localize the interference region between two objects without having to examine each individual voxel.

Using octree shape representation, interference among objects can be detected by traversing their octrees in parallel [13]. Octrees can represent the details of object shapes to different degrees using different numbers of levels, but the computational cost is proportional only to the actual number of nodes visited. Even an octree with limited depth is useful for approximately identifying object parts causing interference. A method in [14] is efficient in environments where there are multiple independently moving objects. In this algorithm, However, the motion of objects is restricted to translation, and there is an overhead for updating positions of and detecting interference among octree shape representations even when the distances among objects are large enough.

In this paper, we propose a method for detecting potential collisions among faces of objects with arbitrary motion (translation and rotation). This method satisfies the requirements of "accuracy" and "computational efficiency" so that the operator will not feel any sense of incompatibility in a virtual workspace. Experimental results show that the proposed method has goog efficiency when there are multiple complicated objects in the environment.

2 Operator Assistance in Virtual Workspace

2.1 Virtual Cooperative Workspace

A virtual cooperative work environment created by computer graphics and having appropriate user interfaces can be shared by multiple users. Figure 1 shows an example of such a virtual cooperative workspace. Each user's viewing position, hand position and hand shape are measured using the system; therefore, the user can directly grasp, move or release objects created by stereoscopic computer graphic images. A user at one site is connected with other users at remote sites through communication networks. The result is a shared virtual workspace, where the users manipulate objects cooperatively.

Let us enumerate the features of a virtual cooperative workspace. First, there exist n objects in the workspace, where $n \geq 2$. Second, there are m $(1 \leq m \leq n)$ moving objects manipulated individually by m operators. Here, each operator is assumed to manipulate only one object, and each object is manipulated by at most one operator. Third, since the objects are moved by hand, the trajectories of moving objects can not be analytically described. Furthermore, the motion of objects may suddenly change. Fourth, data on the motion of objects is extracted at discrete time instants restricted by the computing time for hand gesture recognition, image generation, etc.

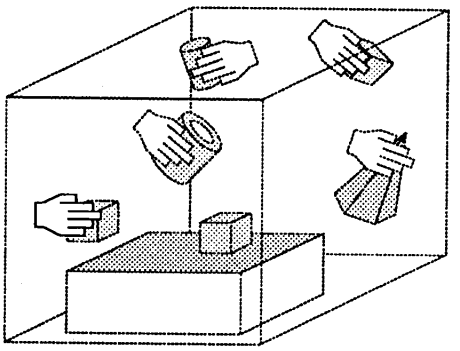


Figure 1: A virtual cooperative workspace

2.2 Operator Assistance

We consider assistance in a task in a virtual cooperative workspace; e.g. changing a layout, matching parts, and so on. In Figure 1 for instance, we consider piling objects on other objects or inserting objects into other objects to support the task. A simplified example is shown in Figure 2. The task involves cooperative work to place object A (operated by a user) onto the upper surface of object B (operated by another user), and avoid collision with standing obstacle C. Without assistance tools, the users may feel this task to be too difficult, i.e. putting two faces (a, b) in contact. However, with assistance from a force feedback device, or a guidance software (described earlier), for example, the users should feel more comfortable. In any case, it is important to select the attracting faces and test the collisions among them efficiently.

When we consider the application of a collision detection method to real-time operator assistance in a virtual cooperative workspace, investigation into the computation time is necessary so that the operator will not feel any sense of incompatibility. In the above example of Figure 2, until the pair of faces (a, b) is found, the operator will move the object directly with his hand. Therefore, the collision/interference test among objects must be performed in real time. However, once the attracting pair of faces is detected, the computer system will assist in the precise movements of objects. Taking a somewhat longer computation time may be tolerated because the operator will not move the object directly in this stage.

Using such an adequate operator assistance system, a user can move a virtual object to any approximate position without any operator assistance and with quick interference tests. Once the object moves close enough to another object, the motion of this object will be supported by one of many assistance tools. Even if this process requires a lot of

computation, this stage is performed by the system with no interaction between users.

Though it is possible to use any collision detection method in a simple environment, this is not so in an environment in which there are multiple independently moving complicated objects. In the following sections, we describe the features of some methods for detecting interference and collisions among objects.

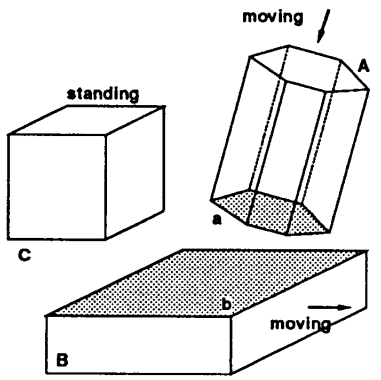


Figure 2: Assistance in a virtual cooperative workspace

3 Shape Representations

3.1 Polyhedral Shape Representation

Polyhedral shape representation is one of the most common shape representations. Interference between polyhedral objects is detected by testing all combinations of faces and edges. The average time complexity for the test (for n objects) is $O(n^2 \cdot EF)$, where E, F are the number of edges and faces in the average object. The computational cost is quadratic in E or F, and also quadratic with respect to the number of objects, n. Therefore, it is difficult to use such a method when there are multiple, independently moving complicated objects. However, if the number of faces or edges that are likely to collide can be restricted, these interference detection methods are useful.

In order to reduce the number of faces and edges, the approximation of objects with bounding spheres or boxes is sometimes used. The interference of two objects is easily detected by comparing the distance between their centers of gravity with the sum of the sizes (radii in the case of spheres) of the approximate shapes. However, the approximation error increases when the objects are complex or concave, and it is not possible to identify directly the attracting parts of the objects (e.g. edges or faces) where collisions occur.

3.2 Octree Shape Representation

The octree represents an object shape by recursive subdivision of a space into octants. A tree node is labeled black (white) if it is completely contained within the object (free space); otherwise, the node is labeled gray.

Using octree shape representation, the interference between two objects can be detected by traversing two trees in parallel [13]. Let N_A, N_B denote a pair of corresponding nodes at any time during the traversal. If either N_A or N_B is white, neither of their children are traversed, and the traversal of the remaining nodes is continued. If both N_A and N_B are black, these nodes are determined to be interfering. If either N_A or N_B is gray, their children are traversed. The depth of the traversal along each path down from the root is determined by the shallower of the two paths terminating in a white leaf. For n objects, a similar traversal of n trees is needed.

Since the time complexity of interference detection using octree shape representation is proportional to the actual number of nodes visited, the average time complexity for detecting interferences among n objects is $O(Kn)$, where the average number of nodes in a tree is K. It is convenient to use octree representation with an adequate level for identifying object parts causing interference.

4 Coarse-to-Fine Collision Detection

The above discussion shows that it is difficult to achieve both "accuracy" and "computational efficiency" by using only one method of collision detection. An efficient method is to first perform an approximate test to identify interfering objects in the entire workspace, then to perform a more accurate test to identify the object parts causing interference/collisions. We propose such a method using bounding shapes, and octree and polyhedral representations of object shapes. The algorithm is described below.

4.1 Outline

Figure 3 shows the control flow in our method. Suppose there are n objects in the workspace, and that for each object all the bounding sphere (box), octree and polyhedral representations are known.

The bounding spheres (boxes) for each object are updated periodically (at discrete time instants $\cdots$, t_{i-1}, t_i, t_{i+1}, $\cdots$) using the observed object motion parameters. When no interference among bounding spheres (boxes) can be found, these motion parameters are stored. Once interference is found, both the octree and polyhedral shape representations for each object are updated using the stored motion parameters. Then, the interference among

objects detected above is tested using octree shape representations. If any interference nodes are found, the faces inside of these nodes are tested for collisions. To avoid collisions, future object positions and orientations are extrapolated using these motion parameters. Potential collisions are then checked using the extrapolated representations to detect collisions between the time interval t_i and t_{i+1} using the following steps.

Here, both the octree and polyhedral shape representations are updated at every time instant. Note that updating polyhedral representations is relatively straightforward. An algorithm capable of updating the octree of a three-dimensional object for arbitrary rotation and translation is proposed in [15].

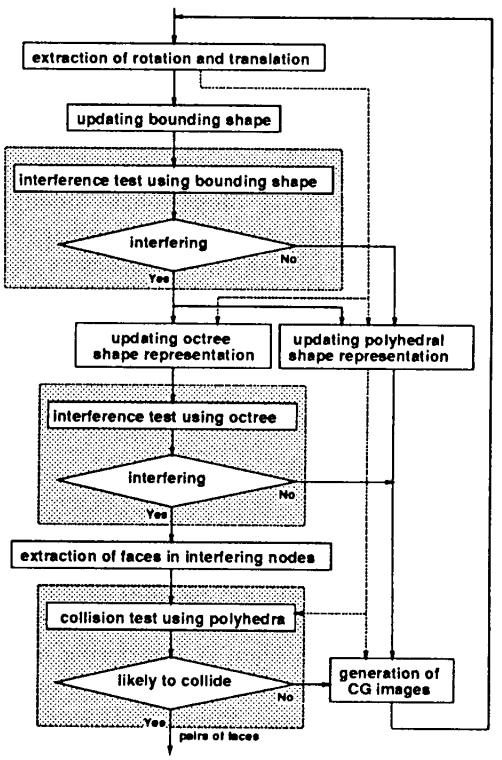


Figure 3: Proposed method of collision detection.

4.2 Procedure

We assume that the speeds of moving objects are slow compared with the sampling intervals. We also assume that the objects are rigid, and their motions between successive time instants are translations and rotations.

4.2.1 Approximate Interference Detection using Bounding Shapes

An approximate interference test in the entire workspace is performed to select close enough objects by using their bounding shapes. We choose these bounding shapes (bounding spheres or boxes) depending on the shape of the objects so as to obtain the smaller approximate error. In the case of bounding spheres, the interferences of objects are detected by comparing the distance between their centers of gravity with the sum of the radii of the bounding spheres. On the other hand, the interferences among bounding boxes (rectangular parallelopipeds) are detected by testing if one of the following positional relationships of all combinations of faces and edges exists: both endpoints of an edge lie on the same side of the plane containing the face (Edge 1), an edge intersects the outside of the face of the plane (Edge 2), or an edge intersects the inside of the face of the plane (Edge 3). We can detect interference for Edge 3.

4.2.2 Interference Detection using Octrees

Until the interferences among bounding shapes are found, the octree and polyhedral shape representations of objects are not updated. Only the matrices of consecutive motions are updated in this period. Once the interferences are found, motions are transformed into a displacement with respect to the initial positions of the octree and polyhedral shape representations.

Interference in the entire workspace is detected by traversing the updated objects' octrees in parallel. If there exists a black node whose corresponding node in another tree is also black, these nodes are considered to be interfering. The traversal is performed down to a predetermined lowest level. At this lowest level, for safety, any pair of corresponding gray nodes is considered to be interfering. After the traversal is completed, all interfering nodes and corresponding objects are identified. This determines the objects and their approximate parts that are likely to collide in the near future.

4.2.3 Extraction of Faces in Interfering Nodes

We extract the faces of an object that intersect with its octree nodes (marked as interfering with other objects). For each of such interfering nodes in an object's octree, the coordinates of the eight vertices of the corresponding cube C are substituted in the equation of the plane T of each face F of the interfering object. If all vertices do not lead to the same sign (positive or negative) for the value obtained after substitution, then this face F is judged as possibly causing the interference detected by octree representation. To determine if the face actually does cause the interference, the polygon S of the intersection between C and T is found. A simple two-dimensional interference detection is then done between S and F. If an intersection is found, F is labeled as interfering; otherwise, F is labeled as noninterfering.

4.2.4 Accurate Collision Detection Using Polyhedra

The faces identified above are checked for collisions. At any time instant t_i, in order not to miss the collisions between time intervals, the possibility of collision between t_i and t_{i+1} is tested by considering the volume expected to be swept by each face during the interval $[t_i, t_{i+1}]$ (see Figure 4). To be conservative, collision is assumed if these volumes intersect even though such intersections are a necessary, but not sufficient, condition for the occurrence of collisions.

For each moving face A, we compute the convex hulls $V_A^{t_i}$ of a set of vertex points of A^{t_i} (i.e. $a_0^{t_i}$, $a_1^{t_i}$, $a_2^{t_i}$,...) and $A^{t_{i+1}}$ (i.e. $a_0^{t_{i+1}}$, $a_1^{t_{i+1}}$, $a_2^{t_{i+1}}$,...) (chapter 3 in [16]) which are expected to be swept by face A during the interval $[t_i, t_{i+1}]$. For each face B^{t_i} with which intersection of A^{t_i} is to be tested during the interval $[t_i, t_{i+1}]$, the convex hulls $V_B^{t_i}$ of a set of vertex points of B^{t_i} and $B^{t_{i+1}}$ are computed. Here, face A and face B at time $t = t_i$ are specified by A^{t_i} and B^{t_i}, respectively.

Then the intersection between $V_A^{t_i}$ and $V_B^{t_i}$ is tested. The intersection is detected by testing whether one of the following positional relationships of all combinations of faces and edges exists: both endpoints of an edge lie on the same side of the plane containing the face (Edge 1), an edge intersects the outside of the face plane (Edge 2), or an edge intersects the inside of the face plane (Edge 3). We can detect an intersection in the case of Edge 3.

This identifies all pairs of faces that are expected to collide in the time interval $[t_i, t_{i+1}]$ by testing for collisions among faces extracted in each node of the octree for which interference has been found. This method is not efficient when the number of vertexes of each face is large. In this case, a more efficient method (such as Muller-Preparata's method in chapter 7 of [16]) might be useful to test for the intersection of convex polyhedra. Figure 4 shows the simplest case (triangles).

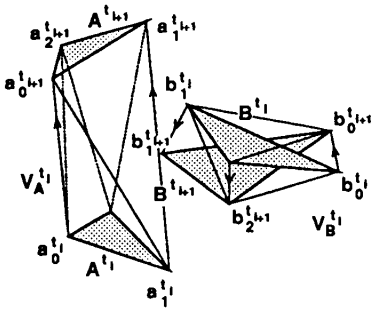


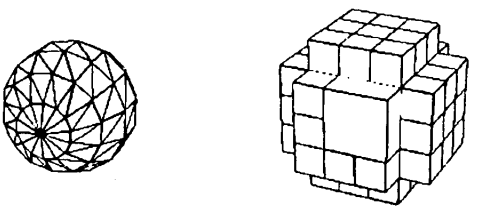
Figure 4: Collision detection between moving faces identified by octree representation as potentially colliding.

5 Experiments

This section first evaluates the performance of the proposed collision detection algorithm using a standardized environment. Then it gives experimental results from practical conditions including general objects.

5.1 Performance Evaluation using Standardized Objects

For performance evaluation, we use sphere-like objects represented by triangular patches. Spheres are selected because of their orientation invariance. An octree shape representation of this object is shown in Figure 5. Here, the octree representation hierarchy is considered to have five levels, and the root node of the octree corresponds to the entire workspace.



(a) polyhedra (b) octree

Figure 5: Experimented object represented with polyhedra (168 triangular faces) and corresponding octree shape representation.

5.1.1 Experiments with Multiple Moving Objects

The interference and collision detection among multiple identical objects (spheres) is tested. Each sphere has a corresponding octree shape representation like Figure 5(b). Constant translation and rotation parameters are given to each object A, B, C, D, ..., and only objects A and B collide. The initial position of the centers and motion vectors of each object are listed in Table 1. The initial positions of six objects (from A to F) in the experimented workspace are shown in Figure 6. All objects are located sufficiently away from each other. Here, the units are equal to the length of the side of the smallest octree cube (or voxel), i.e. level 0 (depth 5). The workspace is divided into M^3, $M = 2^5$ voxels at the lowest level. The diameter of the polyhedral shape representation of each sphere is 3.8 voxels with respect to 5-level octrees, while the diameter of the octree shape representation of each sphere is 5 voxels. The size of the bounding shape is slightly larger than its octree shape representation.

In the beginning, after the positions of the bounding shapes of each object have been updated, an approximate test is performed to identify close objects in the entire workspace at every processing cycle. Once the intersections among bounding shapes

Table 1: Initial positions and motions of objects (unit: 5-level octree)

object	initial positions (*voxels*)	translation (*voxels/cycle*)	rotation (*degrees/cycle*)
A	(4.0, 4.0, 4.0)	(0.03, 0.015, 0.02)	(0.0, 0.05, 0.05)
B	(18.0, 12.0, 14.0)	(-0.03, -0.015, -0.02)	(0.05, 0.05, 0.0)
C	(25.0, 25.0, 14.0)	(0.005, -0.01, 0.005)	(0.0, 0.05, 0.05)
D	(14.0, 3.0, 25.0)	(-0.01, -0.005, 0.0)	(0.0, 0.05, 0.05)
E	(3.0, 25.0, 3.0)	(0.01, 0.0, 0.01)	(0.0, 0.05, 0.05)
F	(25.0, 14.0, 25.0)	(-0.005, -0.001, -0.001)	(0.0, 0.05, 0.05)

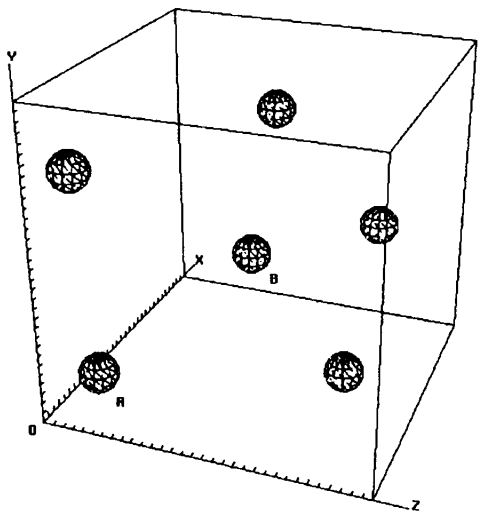


Figure 6: Experimental space including six objects (identical spheres) at initial positions

are found, the positions of the octree and polyhedral shape representations of each object are updated, and a test is performed to identify interfering object parts using the octree representations of the object shapes. If any interfering nodes are found, the polyhedral representations of the object shapes are used to more accurately identify the object parts causing the interference and collisions. We measure the computation time of each processing cycle on a workstation (Silicon Graphics ONYX).

One of the results from two moving objects having 168 triangular patches and 5-level octree shape representations is shown in Figure 7. Both cases using bounding spheres and bounding boxes were experimented. For the bounding boxes, up to time $t = 104$ (*cycles*), no interference can be found; therefore, the computation cost is relatively small (less than 1 msec: minimum measurable resolution time in our experiment). Once the intersections among bounding boxes are found at $t = 105$, more computation is needed to update the positions of the octree and polyhedral shape representations of each object, and to test the interference among identified objects using the octree representations (Note that even this stage requires about 25 msec.) At $t = 169$, interfering nodes of octree are found, and much more computation is needed to detect faces in the interfering nodes and to test the collisions among them. Though interfering nodes are found, the detected faces do not collide up to $t = 196$. Finally, at $t = 196$, collisions among detected faces are found, and this experiment is terminated. At the last stage of this collision detection, 299 msec is needed to identify 8 pairs of faces that are going to collide from 37 interfering octree nodes. Here, we test the collisions among faces identified by each node of the octree for which interference has been found.

For the bounding spheres, the shape of the graph is quite similar except when the intersections among bounding spheres are found ($t = 144$). The computation time for the intersection test of the bounding rectangular parallelopiped is almost the same as that for the bounding spheres.

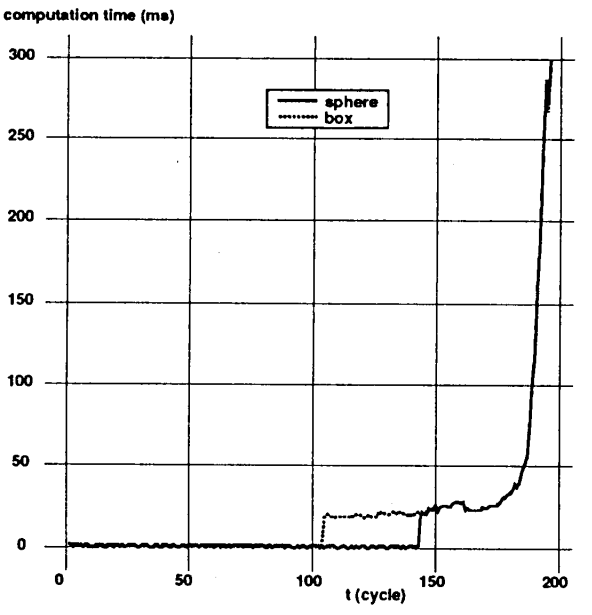


Figure 7: Computation time for each processing cycle of collision detection between two same objects having 168 faces by the proposed methods (using bounding spheres and boxes)

Experimental results obtained using bounding spheres for multiple moving objects are shown in Figure 8. In this figure, the computation time of each processing cycle for collision detection among multiple identical objects having 168 faces is measured.

The computation time of updating the bounding shapes, and octree and polyhedral shape representations are almost equal against the increase of the number of objects. In addition, in the case of bounding boxes, we can get almost the same results. In the beginning period, the computation time is almost 1 msec for six objects for both bounding shapes.

In comparison with a conventional collision detection algorithm using only polyhedral shape representation (which is described in 3.1), our method accomplishes the collision test very quickly in any stage. For example, the computation time for the polyhedral method requires about 29 seconds for two objects. The polyhedral method needs to test collisions among all combinations of faces at every processing cycle; therefore, considerable computation is necessary throughout the experiment. For collision detection among spheres, constraints of parametric methods might be useful in eliminating the candidate faces. However, since these types of constraints are not always good for concave or complex objects, we compared our proposed method with the polyhedral representation collision detection algorithm, which does not use the above constraints.

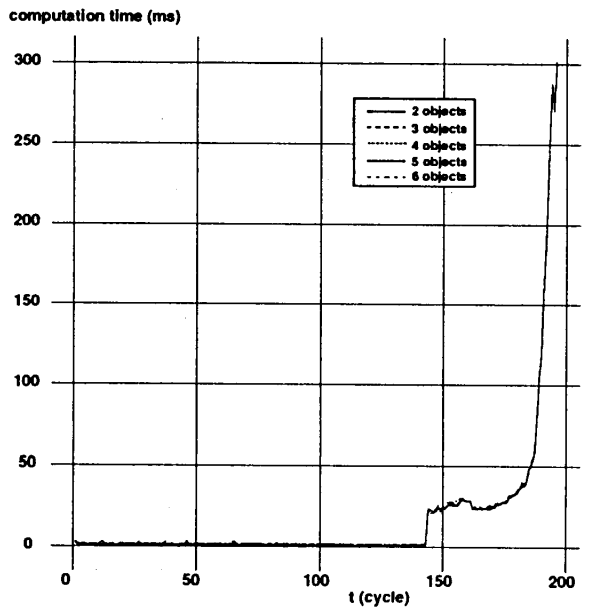


Figure 8: Computation time of each processing cycle for collision detection using bounding spheres among multiple identical objects

In Figure 9, the computation time of the proposed method at the last stage of collision detection which requires maximum computation is compared with the collision detection method using only polyhedral shape representation, for various numbers of objects. The experimental results show the efficiency of the proposed method when the number of objects increases.

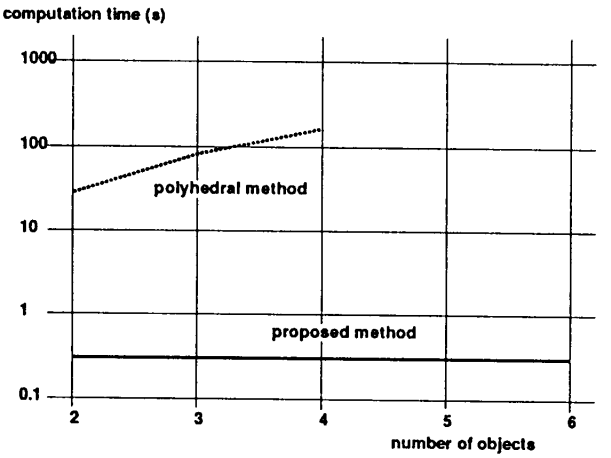


Figure 9: Computation time of each processing cycle for collision detection among multiple moving objects having 168 faces against the number of objects

5.1.2 Experiments with Two Moving Objects Having Different Numbers of Faces

The interference and collision detection between two identical objects (spheres) represented by several kinds of polyhedral shape representations, each having a different number of planar patches is tested. Figure 10 shows these results using bounding spheres. In this figure, the computation time at the last stage of the proposed collision detection, which requires maximum computation, is compared with that of the conventional collision detection method without any constraint of bounding shapes and octrees, i.e., using only polyhedral shape representation. The experimental results show the efficiency of the proposed method when the number of faces of the objects increases.

5.2 Experiments for Practical Environment

Our proposed algorithm is applied to a practical environment. A space shuttle, a statue of venus, and a chair, each represented by triangular patches and an octree shape representation are used. The octree shape representation of each object is generated in advance by the method of [17] from its polyhedral shape representation (see Figure 11 for space shuttle example).

5.2.1 Space Shuttles

The interference and collision detection among six identical objects (space shuttles) is tested with their bounding spheres. The initial positions and directions of the objects (from A to F) in the experimental workspace are shown in Figure 12. Constant translation and rotation parameters are given to each object and only objects A and B collide. Here,

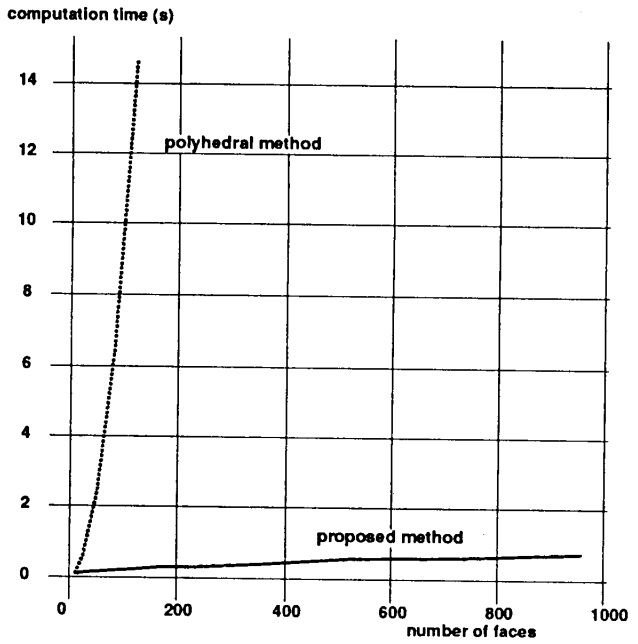
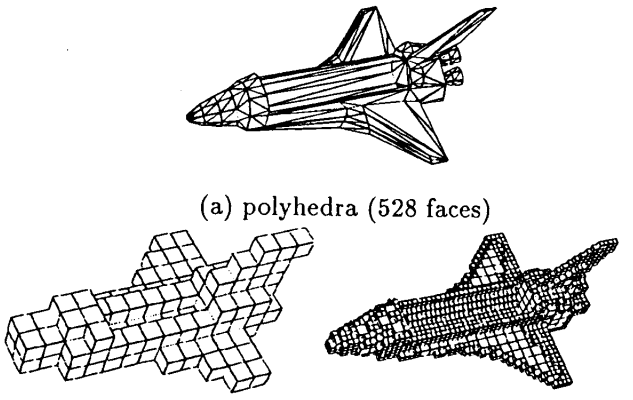


Figure 10: Computation time of each processing cycle for collision detection between two identical objects against the number of planar patches of the objects.

level-6 octree is used. In this experiment, the faces of the polyhedral shape representations are all triangles, and the octree shape representation of Figure 11 (b) is used by integrating its octree root node into the level-2 (depth 4) octree node whose root node corresponds to the entire space.



(a) polyhedra (528 faces)

(b) octree (level 4)

(c) octree (level 6)

Figure 11: A space shuttle represented using polyhedra and its corresponding octree shape representation.

The results from the six moving objects (space shuttles) each having 528 triangular patches are shown in Figure 13. In this case, the environment is considered to have 3168 faces in total. Up to time $t = 115$ (*cycles*), no interference can be found; therefore, the computation cost is relatively small (less

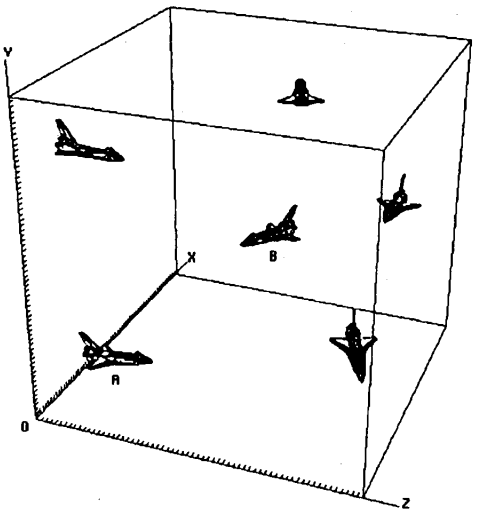


Figure 12: Experimental space including six objects (space shuttles) at their initial positions.

than 1 msec: minimum measurable resolution time in our experiment). Once the intersections among bounding shapes are found at $t = 116$, more computation is needed to update the positions of the octree and polyhedral shape representations of each object, and to test the interference among identified objects using the octree representations. At $t = 188$, interfering nodes are found, and much more computation is needed to detect faces in the interfering nodes and to test the collisions among them. Though interfering nodes are found, the detected faces do not collide up to $t = 183$. Finally, at $t = 184$, collisions among detected faces are found, and this experiment is terminated. At the last stage of this collision detection, 263 msec is needed to identify 4 pairs of faces that are going to collide from 45 interfering octree nodes. As easily understood from the result in Figure 8, results from other object numbers are quite similar to Figure 13's.

On the other hand, the computation time for collision detection for six objects without using octree, i.e., using only polyhedral shape representation, is 70 minutes. while our proposed method requires 263 msec. In this case, our proposed method detects collisions in about 0.06% of the time required by the polyhedral method.

5.2.2 A Cluttered Environment

An environment that includes several kinds of objects is tested (Figure 16). Here, a space shuttle, a statue of venus and a chair having 528, 1868, and 146 faces, respectively, are used. Each object has a corresponding octree shape representation as shown in Figure 11 or 15. Many types of collisions are observed by giving different sets of translation and rotation parameters to each object. Figure 17 shows one of the results from the collision between

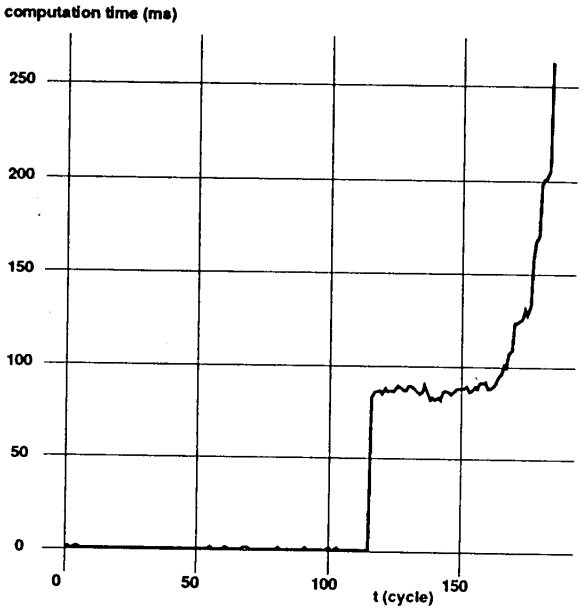


Figure 13: Computation time of each processing cycle for collision detection among six objects (space shuttles).

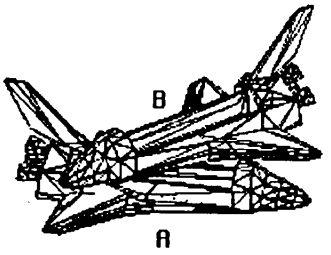


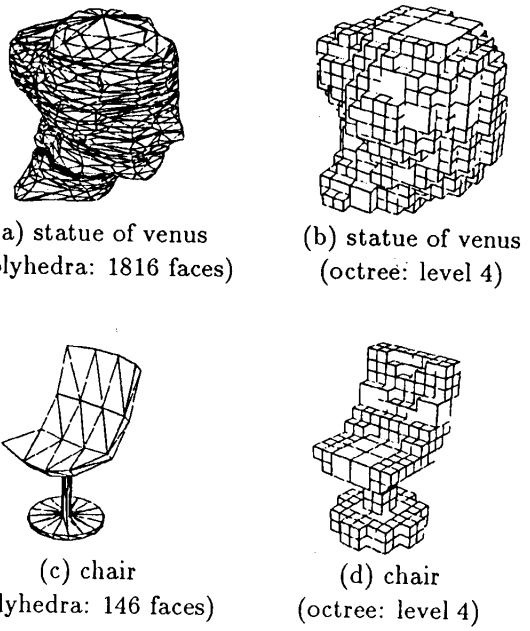
Figure 14: A snapshot of an instant in which collision is detected.

the venus statue and the space shuttle. At the last stage of this collision detection, 1132 msec is needed to identify 2 pairs of faces that are going to collide from 48 interfering octree nodes, while the polyhedral method without using octree requires 27 minuites for the same environment.

6 Application to Operator Assistance in a Virtual Cooperative Workspace

As described above, some pairs of faces that are likely to collide are detected by our proposed collision detection algorithm. Using the detected pairs of faces we can establish the operator assistance system in a virtual cooperative workspace.

In the example of Figure 2, however, not only the pair of face a of object A and face b of object B, but also some pairs that include the side faces of object



(a) statue of venus (polyhedra: 1816 faces)

(b) statue of venus (octree: level 4)

(c) chair (polyhedra: 146 faces)

(d) chair (octree: level 4)

Figure 15: Examples of experimental general objects represented using polyhedra and their corresponding octree shape representation.

A might be detected. Since the way to handle these pairs of faces depends on the task to be assisted and the assistance methods, it should be considered independently of collision/interference detection methods. In the case of Figure 2, if we assume the task to be assisted as cooperative work, such as putting object A operated by a user onto the upper surface of object B operated by another user and avoiding collision with standing obstacle C, we can detect the pair of attracting faces (a and b) by using such properties as the parallelism of a pair, the time of collision, the area of faces, or the horizontality of faces. Once the attracting faces are selected, such tools as a force feedback device, which can generate a reaction between two faces touching each other[2], and guidance software which can lead objects to appropriate positions [3] are available. Figure 18 shows the concept of the latter method. With this tool, when the operator grasps an object in the virtual environment and moves it close to another object (i.e. within a threshold), both objects automatically become attached. In other words, all the operator needs to do is to control two translations and one rotation of freedom of the grasped object.

A user can move a virtual object to an approximate position without any operator assistance. Here, quick interference tests are executed. Once the object moves close enough to another object, the motion of this object will be supported by one of the assistance tools. Even if this process requires much computation, this final stage is necessary for an accurate test. Even for a complicated environment like

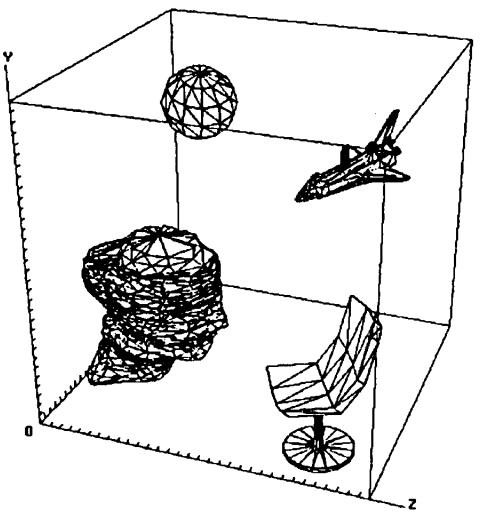


Figure 16: Experimental space (level-5 octree is used) including several kinds of objects

Figure 14, our proposed collision detection method requires 263 msec to identify pairs of faces that are going to collide. The method is therefore sufficient for real-time applications with operator assistance.

Our proposed collision detection method is efficient compared with polyhedral methods, however, it is not sufficient for real-time applications in very complicated situations like in Figure 16. Since the proposed collision detection method can be easily implemented on a parallel processor, real-time collision detection in environments containing multiple independently moving complicated objects, can be achieved. Concretely speaking, the computation time of collision detection at the last stage which requires maximum computation can be reduced by testing the collisions between pairs of faces parallelly.

7 Summary and Conclusion

We have proposed a method for detecting potential collisions among faces of objects. which satisfies "accuracy" and "computational efficiency". Experimental results have shown the efficiency of the proposed method, especially when there are multiple, complicated objects with arbitrary motion in the environment. The method tests collisions in the entire workspace all the time, and achieves an appropriate efficiency in detecting pairs of faces about to collide. These detected pairs of faces are used for the operator assistance method.

Collision detection is accomplished in about 1 msec when an object is sufficiently away from another object in our experiments, and even in the last stage of accurate collision detection, it requires only 263 msec to identify the pairs of faces that are likely to collide.

This collision detection method enables the user

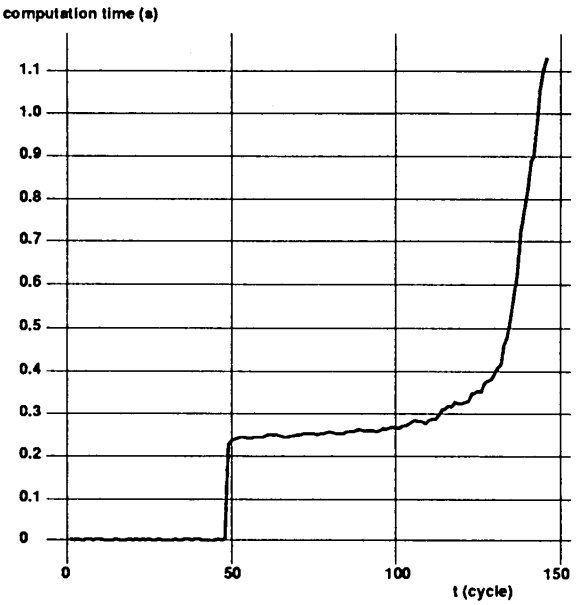


Figure 17: Computation time of each processing cycle for collision detection among objects (a space shuttle, a statue of venus and a chair) in a practical environment.

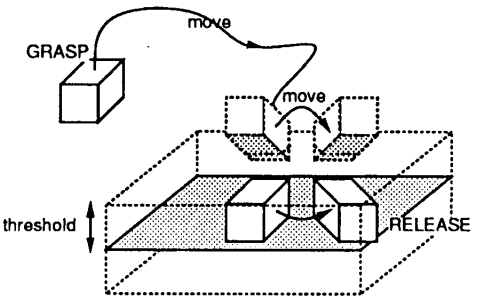


Figure 18: An example of operator assistant

in a virtual workspace to manipulate a virtual object without any sense of incompatibility while he moves the object directly with his hand. Once the object moves close enough to another object and the attracting pair of faces is detected, the motion of the object is supported by assistance tools; thus, the precise placement of the object is possible. Taking a somewhat longer computation time may be tolerated because the operator will not move the object directly in this stage.

References

[1] Takemura, Haruo and Kishino, Fumio. Cooperative work environment using virtual workspace. In *CSCW*, pp. 226–232, 1992.

[2] Iwata, Hiroo. Artificial reality with force-feedback: development of desktop virtual space with compact master manipulator. *Computer Graphics*, Vol. 24, No. 4, pp. 165–170, 1990.

[3] Chanezon, A., Takemura, H., Kitamura, Y., and Kishino, F. A study of an operator assistant for virtual space. In *Virtual Reality Annual International Symposium*, pp. 492–498. IEEE, 1993.

[4] Hwang, Yong K. and Ahuja, Narendra. Gross motion planning – a survey. *ACM Computing Surveys*, Vol. 24, No. 3, pp. 219–291, 1992.

[5] Boyse, John W. Interference decision among solids and surfaces. *Communications of the ACM*, Vol. 22, No. 1, pp. 3–9, 1979.

[6] Canny, John. Collision decision for moving polyhedra. *IEEE Trans. on PAMI*, Vol. 8, No. 2, pp. 200–209, 1986.

[7] Kawabe, S., Okano, A., and Shimada, K. Collision detection among moving objects in simulation. *Robotics Research*, Vol. 4, pp. 489–496, 1988.

[8] Moore, Matthew and Wilhelms, Jane. Collision detection for computer animation. *Computer Graphics*, Vol. 22, No. 4, pp. 289–298, 1988.

[9] Sclaroff, Stan and Pentland, Alex. Generalized implicit functions for computer graphics. *Computer Graphics*, Vol. 25, No. 4, pp. 247–250, 1991.

[10] Hubbard, Philippe M. Interactive collision decision. In *Symposium on Research Frontiers in Virtual Reality*, pp. 24–31. IEEE, 1993.

[11] Youn, Ji-Hoon and Wohn, K. Realtime collision decision for virtual reality applications. In *Virtual Reality Annual International Symposium*, pp. 415–421. IEEE, 1993.

[12] Garcia-Alonso, A., Serrano, N., and Flaquer, J. Solving the collision decision problem. *Computer Graphics and Applications*, Vol. 14, No. 3, pp. 36–43, May 1994.

[13] Ahuja, N., Chien, R. T., and Bridwell, N. Interference detection and collision avoidance among three dimensional objects. In *International Conference on Artificial Intelligence*, pp. 44–48, 1980.

[14] Kitamura, Y., Takemura, H., Ahuja, N., and Kishino, F. Efficient interference detection among objects using octree and polyhedral shape representation. In *Asian Conference on Computer Vision*, pp. 775–779, 1993.

[15] Weng, J. and Ahuja, N. Octree of objects in arbitrary motion: Representation and efficiency. *Computer vision, graphics, and image processing*, Vol. 39, No. 2, pp. 167–185, 1987.

[16] Preparata, Franco P. and Shamos, Michael Ian. *Computational geometry, an introduction.* Springer-Verlag, 1988.

[17] Kitamura, Y., Bayle, M., Takemura, H., and Kishino, F. Generation of an octree from a polyhedral shape representation. In *IEICE Conference 1994*. D-604, 1994. (in Japanese).