Proceedings of the 2006 IEEE/RSJ
International Conference on Intelligent Robots and Systems
October 9 - 15, 2006, Beijing, China

Real-time and Accurate Multiple Contact Detection between General Curved Objects

Wusheng Chou
IMI lab, Dept. of Computer Science
University of North Carolina - Charlotte
Charlotte, NC 28223, USA
wschou@uncc.edu

Jing Xiao
IMI lab, Dept. of Computer Science
University of North Carolina - Charlotte
Charlotte, NC 28223, USA
xiao@uncc.edu

Abstract – Contact detection based on computing minimum distance is a fundamental issue important to many applications. A largely unsolved problem is how to detect multiple contacts that are formed simultaneously between non-convex and non-polyhedral general objects both accurately and in real-time. This paper presents an effective solution to the problem. Our approach first locates the pairs of closest components by fast intersection checking based on hybrid bounding volume hierarchies of surface components. For each pair of such components, it then finds the pairs of closest points and corresponding pairs of closest parametric features by combining collision detection or minimum distance query between polygonal meshes of those components and exact distance computation between the parametric features. Implementation results show that this approach can compute multiple simultaneous contacts between general objects both very accurately and efficiently in the order of several milliseconds regardless of the numbers of features on the objects.

Index Terms – multiple contact detection, curved objects, minimum distance computation, collision detection, real time

I. Introduction

The information of contact states between two objects is essential to many robotics and haptics applications, such as robot motion planning [1], virtual assembly prototyping [2], maintenance verification [3], virtual manipulation [4], haptic rendering and dynamic simulation, etc. Often it is not enough to know just whether two objects are in contact or not, but to obtain information such as the *exact* contact points is also needed in *real time*. Because of the stringent requirements on both accuracy and time efficiency, which are often conflicting requirements, the problem is largely unsolved for interacting non-convex and non-polyhedral objects where multiple contact regions can occur simultaneously.

Related existing literature can be classified into the following categories:

- Collision detection based on intersection checking

The main approach in this category represents objects in certain bounding volume hierarchies to facilitate fast intersection checking. Many types of bounding volumes are introduced in the literature, such as spheres [5], oriented bounding boxes (OBBs) [6], axis-aligned bounding boxes (AABBs) [7], discrete orientation polytopes (k-dops) [8], and octagonal bounding volumes [9]. Methods in this category can be fast but with limited accuracy because they cannot provide exact contact points which are important to many applications.

- Distance computation based on polygonal models

There is considerable research on calculating the minimum distance between two objects based on polygonal surface models, which is surveyed in [10] and in [11]. More recent methods such as PQP [14] and SOLID [15] combine convex bounding volume hierarchies and efficient distance computation to deal with concave objects. Reference [15] and reference [16] can only provide a penetration depth for single intersection but cannot deal with multiple intersections. All the above methods cannot handle multiple contacts.

Moreover, because methods in this category use polygonal approximations of objects, it is quite possible that the distance between the polygonal approximations of two curved objects is not equal to zero when two convex surfaces of the actual objects already penetrate each other. For these methods, a trade-off has to be made between the accuracy of the approximation and the efficiency of the overall computation.

- Distance computation based on exact analytic descriptions of curved objects

The methods based on exact analytic descriptions of curved objects avoid the approximation problem of the methods based on polygonal models but also have their own various limitations. Most of the current work mainly focuses on convex curved objects, and real-time computation is seldom achieved. Generally, the problem is treated as root-finding of some equations that describe conditions for minimum distance. The minimum distance query between curved objects is also treated as a problem of solving univariate polynomial equations [17], or a dynamical control problem [18]. Analytic and iterative numerical methods are often used to solve for solution(s). The analytic approach guarantees to find all extrema of the distance function but cannot deal with complex objects, especially when the objects are in motion. Local numerical optimization methods also need adequate starting points in order to converge to good solutions. References [19][20][21][22] are quite efficient but at the cost of accuracy.

Multiple occurrences of minimum distance or multiple contacts between different pairs of features are handled in [23] with great accuracy. However, the overall performance of the approach depends on the total number of feature pairs tracked.

- Hybrid methods

1-4244-0259-X/06/$20.00 ©2006 IEEE

There are approaches trying to combine the methods based on polygonal models and those based on analytic models of objects [24] [25] [26]. In [24], once an intersection between two objects in polygonal models are detected, the two objects are pulled back to a contact state based on computation of penetration distance between parametric surfaces with local numerical techniques. This method does not seem to consider the inter-dependence among multiple contact regions and may not always find the correct contact state. In [25], polyhedra are used to approximate curved objects with convex surfaces, but for concave surfaces, computation is done directly on the algebraic surface models based on Gauss-Newton method. Because it is time consuming to obtain the initial values of the variables for such computation especially when objects are complex and are in motion, real-time can hardly be achieved. Reference [26] improves the process of finding initial values and used the method in [20] to find the approximate minimum distance. However, none of the above methods is able to handle simultaneous, multiple occurrences of multiple contacts. They cannot provide exact information of such contact situations with both accuracy and real-time efficiency.

This paper aims to fill the gap by introducing an approach that can detect multiple contacts between general curved objects both exactly and in real-time. Our approach extends the advantages of bounding volume hierarchies, collision detection and distance computation based on both polygonal and analytic surface models of objects while avoiding their respective drawbacks by providing key "bridges" that integrate, mend, and extend partial solutions obtained from these existing algorithms.

The rest of this paper is organized as follows: Section II gives the overview of our approach. Section III presents the object models used by our approach. Section IV discusses the scope of our algorithm. We present and discuss the implementation results in Section V. Section VI concludes the paper.

II. Overview of the Approach

First we define the interaction states between two objects by taking into account digital errors in the following robust way: two objects are in *contact* if the minimum distance d between them is within a small range:

$$0 \le \delta_1 \le d \le \delta_2 \qquad (1)$$

if $d < \delta_1$, then the two objects are in *penetration*, and if $d > \delta_2$, then the two objects are *separated*. Our approach to exact contact detection is to find exact contact points based on the above definition.

For object models, we establish bounding volume trees for surface components of general objects and dual representations for each surface component in terms of both polygonal meshes and surface features described parametrically. These representations are built offline during pre-processing.

Our algorithm consists of at most three stages in one time step to detect contacts between two moving objects:

- *Stage 1*: intersection checking between component-based bounding-volume trees

Unlike existing bounding-volume tree approaches using primitive polygons as basic building blocks [6] [7], i.e., each leaf node of a bounding-volume tree contains a polygon, our approach significantly reduces the size of such a hierarchy by making a tree of bounding volumes with object *surface components* (explained in detail in Section III) as primitive building blocks, which are much larger entities than simple polygons. As an object has far fewer number of surface components than the number of polygons in its surface mesh model, our component-based bounding volume tree is much smaller, so that much faster and mesh-size independent checking of potentially closest surface components can be done between two general objects.

Once two leaf-level bounding volumes of the two objects are detected as intersecting, our algorithm enters Stage 2 below.

- *Stage 2*: collision detection or distance computation based on polygonal mesh models.

For each pair of intersected leaf-level bounding volumes of components, take the polygonal mesh models of the corresponding components C_1 and C_2. If the two mesh models of the surfaces do not intersect, but they are close enough in distance, enter Stage 3. In Stage 2, we currently use an extended implementation of the GJK algorithm [12] to calculate the minimum distance and the closest points between two convex polygonal models.

If at least one concave component is involved in a pair of closest components, we first offset the mesh of one concave component by δ to obtain a proxy mesh. Next we use a fast collision detection algorithm to detect all the colliding triangles between the mesh and proxy mesh of the concave component and the mesh of the other component (which could also be concave). Currently we use the OPCODE [28] for this purpose. Our algorithm further detects if there are colliding triangles between the two meshes of the two components. If so, penetration is reported. Otherwise, if the colliding triangles are only between the proxy mesh of one component and the mesh of the other component, a contact state occur. The algorithm then separates the collided triangles into groups, and each group consists of pairs of colliding triangles that form a connected collision region. For each group i ($i \ge 1$), the algorithm computes the exact intersection lines between the intersecting triangles and finds the geometric centre $p_{c,i}$ of the area bounded by the intersection lines. Our algorithm enters Stage 3 at this point.

- *Stage 3*: exact distance computation between exact parametric models of surface features

Use the information obtained in Stage 2 to find the pairs of closest parametric features and the pairs of closest points between the two features.

Let C_1 and C_2 denote a pair of contacting components from Stage 2. If the pair of components C_1 and C_2 are both convex, Stage 2 finds the pair of closest points p_1 and p_2 on the polygonal meshes of C_1 and C_2 respectively. In Stage 3, p_1 and p_2 are used to find the corresponding closest features f and h on the exact models of C_1 and C_2 as well as the starting points s_1 and s_2 for finding the closest points on f and h. We can

decide the closest feature f of C_1 that p_1 corresponds to according to the pre-constructed data structure for p_1.

Next, the starting point s_1 on f can be determined by projecting p_1 on f along the normal direction. The starting point s_2 on h can be decided similarly from p_2. s_1 and s_2 are then input to the Newton's optimization routine to find the exact closest points on f and h.

If C_1 (or C_2) is a concave component, then Stage 2 passes to Stage 3 the points $p_{c,i}$ ($i \geq 1$). Now $p_{c,i}$ is projected to the exact surface models of both C_1 and C_2 along their normal directions to obtain the corresponding closest features f_i and h_i on the exact models of C_1 and C_2. The projected points $s_{1,i}$ and $s_{2,i}$ on features f_i and h_i respectively are next used as the starting points for the Newton's optimization routine to find the exact closest points on f_i and h_i.

Our implementation shows that such a pair of points s_1 and s_2 (or $s_{1,i}$ and $s_{2,i}$) found as described above are very good starting points for further optimization and always result in quick convergence to the actual pair of closest points between the closest features.

Note that there are three possible kinds of output results. If no component bounding volumes or surface meshes are intersecting, the output is "no contact." If at least one pair of surface meshes is intersecting, the output is "penetration." Otherwise, contacts are identified.

III. REPRESENTATIONS OF OBJECTS

The whole surface of a general object consists of surface patches, curves/lines bounding the surface patches and isolated points (i.e., the intersection points of curves). These surface patches, curves/lines and isolated points are the *features* of a curved object, called *faces*, *edges*, and *vertices* respectively. A face is *convex* if, when it is viewed from the outside of the object toward the outward normal of the face, it is convex. Otherwise, a face is *concave*.

We partition the surface of a general object into a minimum number of *surface components*. There can be two types of surface components. A *convex component* consists of connected convex faces of the object that does not form any concavity and is the convex hull of the connected convex faces which have concave edges. A *concave component* consists of concave faces of the object. Thus, if an object is convex, its whole surface is a single convex surface component. If an object is not convex, it has more than one surface component. If an object has concavities formed by convex faces only, it has only convex components. If an object has concavities formed by concave faces, it has concave components. A general object may have both convex and concave components.

On top of the surface components, a hierarchical representation of bounding volumes is constructed as a *component-based bounding-volume tree*. A leaf node of this binary tree contains the bounding volume of a single surface component.

Each component is further represented both in terms of a polygonal mesh model and parametric descriptions of its surface features, which we call *dual representations*. We explain the component-based bounding volume tree and the dual representations of components below in detail. Note that all these representations are constructed in pre-processing.

A. Component-based tree of hybrid bounding volumes

We build a binary tree of bounding volumes for an object in the following way: the root of the tree is a bounding volume of the entire object surface; the surface is then partitioned into two unions of surface components, and the bounding volume of each union is a child node of the root; next the union of components of each child node of the root is again partitioned into two smaller unions of components, whose bounding volumes define the two children of the node, and so on. A leaf node of this binary tree contains the bounding volume of a single surface component.

There is a trade off between the evenness of fit and the time cost of intersection checking between a pair of bounding volumes. Octagonal bounding volumes [9] significantly improve the evenness of fit by cutting off the corners and thus improve the culling rate at the cost of slightly more expensive intersection checking. In our current implementation, we combine the advantages of both OBBs and octagonal bounding volumes by using OBBs in the non-leaf nodes and octagonal bounding volumes in the leaf nodes of our binary tree. The results are quite satisfactory as shown in Section V. Of course, our choice of bounding volumes is not exclusive. Other bounding volumes (such as k-dops [8]) could also be used.

Once a bounding volume is constructed, we can magnify its size to provide some tolerance by a distance δ (see Fig. 1). δ should be larger than the displacement of an object's motion in a single time step. This is to ensure that when two bounding volumes are first detected as intersecting from not intersecting in the previous time step, the actual object components are not in collision, and they can be close enough to be considered in contact (by our definition in Section II).

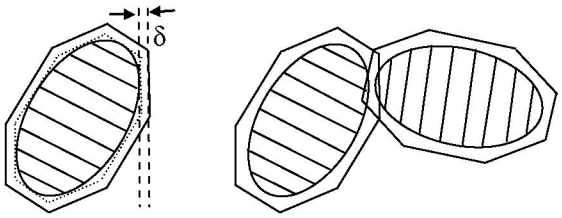

Fig.1. Octagonal bounding volumes with tolerance are intersecting but there is no intersection between the two components.

B. Dual representations of surface components

We obtain a *parametric representation* of a surface component in terms of their parametric features and the adjacency relations among those features. The parametric representation of a face involves two independent parameters u and v, and that of an edge uses one parameter. For a vertex, no parameter is needed.

In addition to the parametric representation of a surface component, we also obtain its polygonal approximation as a mesh of triangles. The initial tessellated polygonal mesh can also be adaptively subdivided based on surface curvature to obtain required approximation accuracy.

For each component, we further establish the correspondences between each vertex and the triangles it belongs to as well as the parametric features by a simple program and store them in a special data structure as the results of preprocessing.

IV. LIMITATIONS OF THE ALGORITHM

Our algorithm is suitable to detect isolated pairs of contact points, which happen most commonly between curved objects, rather than continuous contact regions that happen between two flat faces (as in the case of polyhedral objects).

This algorithm also operates under a common basic assumption (that existing distance computation or collision detection algorithms based on surface representations of objects often assume): the penetration state where one solid object is contained completely inside another solid object so that there is no surface intersection is prevented. This assumption is reasonable in applications such as haptic operations and dynamic simulation with solid objects: in those cases, an object is moved away from penetration by simulating physics.

V. IMPLEMENTATION, TESTING, AND PERFORMANCE

We have implemented our approach in Visual C++ and run it on a Pentium Processor of 2.8GHz CPU with 1GB RAM. Six curved solid objects are used in testing the performance of the proposed approach: the paperweight (3 convex surface components), the goblet-shaped solid (8 surface components), the bowl (2 surface components), the pencil(3 surface components), the solid jar (3 surface components), and the vase(3surface components).

As the first stage of our algorithm, intersection checking is conducted between component-based bounding-volume trees. The running times for intersection checking with the paperweight and the goblet-shaped solid as the two objects are presented in Table 1. The component-based binary bounding-volume tree for paperweight has 3 levels and a total of 5 nodes, and that for the goblet-shaped solid has 4 levels and a total of 15 nodes. Table 1 shows the time of intersection checking between two entire trees, with respect to different numbers of intersecting leaf nodes. The data show that the entire intersecting checking is very fast in the order of hundreds of microseconds (μs).

TABLE 1
Average time (in millisecond) for intersection checking between two bounding volume trees

Pairs of colliding leaf node	0	1	2	3	4	5	6
Average time	0.01	0.15	0.3	0.33	0.42	0.46	0.51

In the following experiments, we test the total running time including Stage 2 and Stage 3 between two objects. Fig.2 shows the average total running time (i.e., including all the three stages) with respect to the above object pairs of different total number of feature pairs: Our approach achieves almost constant time when only one contact occurs. The average running time of our algorithm ranges from 1.4 to 1.8 millisecond.

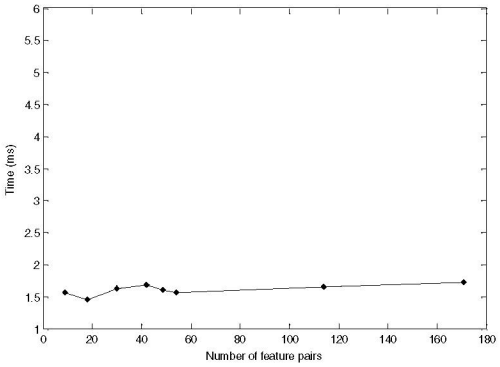

Fig.2. Average total running time vs. number of feature pairs

TABLE 2
Average time (in millisecond) of the Newton method

	Point-curve	Curve-curve	Curve-surface	Surface-surface
Time	0.17	0.32	0.62	0.95

Table 2 shows the average running time for searching the pair of closest points on the pair of closest exact features of two objects based on the Newton method, which ranges from 0.1 to 1 millisecond depending on the type of the feature pair. Clearly, the numerical process sometimes takes a greater part of the total running time.

Fig. 3 shows the multiple collision regions between the mesh model of the pencil and the proxy mesh model of (in grey colour) of the bowl, which means multiple contacts between the pencil and the bowl. In Fig. 3, the intersecting triangles are rendered in red, and the black lines are intersection lines of the intersecting triangles. Fig. 4 shows two contact states between the pencil and the paperweight, and Fig.15 shows two contact states between the paperweight and the goblet-shaped solid. Fig. 6 gives the example of multiple contacts between two vases involving concave surfaces. Table 3 gives the average running time of the entire algorithm including all three stages with respect to contact states where two contacts are detected.

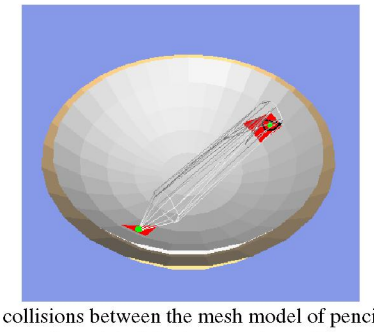

Fig.3 Multiple collisions between the mesh model of pencil and the proxy mesh model of the inner surface of bowl

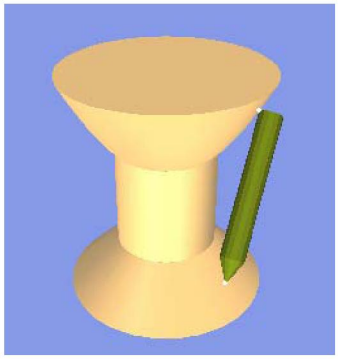

Fig.4. Multiple contacts between the pencil and the paperweight

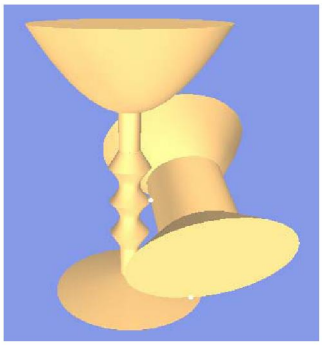

Fig.5. Multiple contacts between the paperweight and the goblet-shaped solid

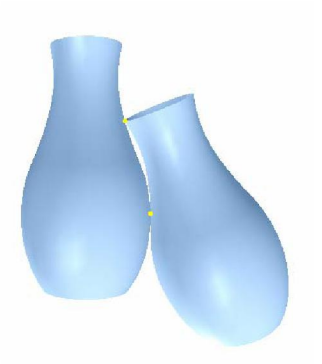

Fig. 6. Multiple contacts between two vases

The time complexity of our algorithm is analysed as follows. In Stage 1 of our algorithm, suppose that the two binary bounding-volume trees have $2n$-1 and $2m$-1 nodes respectively, corresponding to n and m components of two objects[1]. Without losing generality, let $n=m+k$, $k \bullet 0$, then the average complexity of intersection checking between the two trees that results in at most three pairs of intersecting leaf nodes, which correspond to at most nine simultaneous contacts can be computed as $O(\log n)$.

In Stage 2, the average running time is approximately linear to the number of component pairs whose bounding volumes intersect (from Stage 1). In Stage 3, the average running time is approximately linear to the number of closest feature pairs (obtained from the results of Stage 2).

TABLE 3
Average total running time (in millisecond)

	2 contacts
Pencil –paperweight	2 .68
Paperweight-goblet-shaped solid	3.13
Pencil-bowl	2.95
Vase-vase	4.87

VI. Conclusions

We have presented an effective approach to solve a largely unsolved problem: exact and real-time detection of contacts between non-convex general curved objects. Our approach has the following major characteristics:

- It is a unified and general framework for exact contact determination between general curved objects. It provides exact contact points and closest feature pairs, which are important for physically-based applications subject to non-penetration constraints.
- It is capable of detecting the simultaneous occurrence of multiple contacts between two objects, including not only those due to concavities formed by convex surface features but also those due to concave surface features.
- Its performance both in terms of efficiency and accuracy is relatively independent of the total number of surface features of the objects, and therefore the algorithm is robust and highly scalable.

With these characteristics, our method is particularly suited for high-fidelity interactive applications, such as those involving high-fidelity haptic rendering.

An interesting future direction is to extend our approach to deal with curved objects in NURBS representation since NURBS is used widely in most CAD systems.

Acknowledgement

The authors are grateful to the support of National Science Foundation of U.S. under grants IIS-0328782 and EIA-0203146.

References

[1] M. Reggiani, M. Mazzoli, S. Caselli "An Experimental Evaluation of Collision Detection Packages for Robot Motion Planning," *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and System*, Vol. 3, pp.2329 - 2334, 2002

[2] S.M. Mok, C.H. Wu, D.T. Lee, "Modelling Automatic Assembly and Disassembly Operations for Virtual Manufacturing," *IEEE Transactions on Systems, Man and Cybernetics*, Part A, Vol. 31, pp. 223 –232,2001.

[3] A. Gomes de Sa, and P. Baacke, "Experiences with Virtual Reality Techniques in the Prototyping Process at BMW," In F. Dai, editor, *Virtual Reality for Industrial Applications*, *Computer Graphics: Systems and Applications*, Chapter 9, pp. 151-158, 1998.

[4] H. Maekawa, and J.M. Hollerbach, "*Haptic Display for Object Grasping and Manipulating in Virtual Environment*," *Proc. IEEE Intl. Conf. Robotics & Automation*, Leuven, Belgium, May 16-21, pp. 2566-2573,1998.

[1]Note that no two bounding volumes share the same component at the same level of the trees, i.e., there is no overlapping bounding boxes at the same level of trees.

[5] S. Quinlan, "Efficient Distance Compution between Non-convex Objects," *Proc. of IEEE Int.l Conf. on Robotics and Automation*, pp. 3324-3329,1994.
[6] S. Gottschalk, M. Lin, and D. Manocha, "Obb-Tree: A Hierarchical Structure for Rapid Interference Detection," *Computer Graphics, Proc. SIGGRAPH'96*, pp.171–180,1996.
[7] G. Vandenbergen, "Efficient Collision Detection of Complex Deformable Models Using AABB Trees," *Journal of Graphics Tools*, Vol. 2, pp.1-13, 1997.
[8] J. T. Klosowski, "Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs," *IEEE Trans. on Visualization and Computer Graphics*, Vol.1, pp. 21-36, 1998
[9] W.S. Chou, J. Xiao, "A Collision Detection Method for Virtual Manufacturing," *Transactions of North America Manufacturing Research*, pp.319-326, 2004.
[10] P. Jimenez, F. Thomas, and C. Torras, "3D Collision Detection: A Survey," *Computers and Graphics*, 25(2): 269-285, 2000.
[11] M.C. Lin, S. Gottschalk, "Collision Detection between Geometric Models: A Survey," *IMA Conf. on Mathematics of Surfaces*, Vol. 1, pp.602-608, 1998.
[12] E. G. Gilbert, D .W. Johnson, and S. S. Keerthi, "A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space," *IEEE J. Robot. Automat.,* Vol. 4, pp. 193-203,1988
[13] M.C. Lin, and J.F. Canny, "A Fast Algorithm for Incremental Distance Calculation," *Proc. IEEE Int. Conf. Robotics & Automation*, pp. 1008-1014, 1991.
[14] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha. "Fast Distance Queries with Rectangular Swept Sphere Volumes", *Proc. IEEE International Conference on Robotics and Automation* , pp. 3719-3726, 2000
[15] Gino van den Bergen. *Collision detection in Interactive 3D Environments*, published by Morgan Kaufmann Publishers, 2003
[16] Y.J. Kim, M.C. Lin, D. Mancha. " DEEP: dual-space expansion for estimating penetration depth between convex polytopes", *IEEE International Conference on Robotics and Automation*, pp.921-925, 2002
[17] C. Lennerz, E. Sch¨omer. "Efficient Distance Computation for Quadratic Curves and Surfaces," *2nd IEEE Conf. on Geometric Modelling and Processing*, pp. 60-69, 2002.
[18] V. Patoglu, R. B. Gillespie, "Extremal Distance Maintenance for Parametric Curves and Surfaces," *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 2817-2823, 2002.
[19] C. Turnbull and S. Cameron, "Computing Distances between NURBS-Defined Convex Objects," *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 3686--3690, 1998.
[20] D. E. Johnson and E. Cohen, "A Framework for Efficient Minimum Distance Computation," *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 3678--3683, May 1998.
[21] D. E. Johnson and E. Cohen, "Bound Coherence for Minimum Distance Computation," *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp.1843-1848, 1999.
[22] F. Thomas, C. Turnbull, L. Ros, S. Cameron, "Computing Signed Distances between Free-Form Objects," *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 3713-3718, 2000.
[23] Z. Zou, J. Xiao. "Tracking Minimum Distances between Curved Objects with Parametric Surfaces in Real Time," *Proceedings of 2003 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Las Vegas, pp.2692-2698, 2003
[24] J. M. Snyder, "An Interactive Tool for Placing Curved Surfaces without Interpenetration," Proceedings of Computer Graphics, pp. 209-218, 1995
[25] M. C. Lin, D. Manocha, "Fast interference detection between geometric models". The Visual Computer 11(10): 542-561 ,1995
[26] D. D. Nelson, D. E. Johnson, and E. Cohen. Haptic rendering of surface-to-surface sculpted model interaction. In Proc.ASME Dynamic Systems and Control Division, vol. 67, 101–108, 1999.
[27] .S. Cameron, "Enhancing GJK: Computing Minimum and Penetration Distance Between Convex Polyhedra," *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 3112-3117, 1997.
[28] Pierre Terdiman, http://www.codercorner.com/Opcode.htm