IMECE99/DSCD, Proceedings of the ASME
Virtual Environment and Teleoperator System symposium
November 1999. Nashville

DSC-3-1

HAPTIC RENDERING: PRACTICAL MODELING AND COLLISION DETECTION

ZhuLiang Cai
Experimental Robotics
Laboratory (ERL)

John Dill
Graphics and Multimedia
Research Laboratory(GMRL)

Shahram Payandeh
Experimental Robotics
Laboratory(ERL)

ABSTRACT

3D collision detection and modeling techniques can be used in the development of haptic rendering schemes which can be used, for example, in surgical training, virtual assembly, or games. Based on a fast collision detection algorithm (RAPID) and 3D object representation, a practical haptic rendering system has been developed. A sub-system determines detailed collision information. Simulation results are presented to demonstrate the practicality of our results.

1. INTRODUCTION

With multimedia developing through graphical animation, stereo sound and live video imaging technology, a technological revolution is now taking place in which user interaction with computers is mediated by an artificial virtual reality. Virtual reality is enhanced by the addition of communication modalities such as stereo, 3D sound, force/tactile feedback, and even taste and smell. In this paper we focus on aspects of tactile feedback, i.e. haptic feedback.

Haptic, a word derived from the Greek word *haptiesthai*, means touch. A haptic interface is composed of three subsystems: Haptic Rendering, Haptic Device, and Haptic Display.

Haptic Rendering, by simulating the forces generated by contact with a virtual model, provides a means of interrogating the model by touch, an additional means of interacting with a virtual environment.

Haptic Devices are the physical means of providing this communication. Such devices should have the dynamic range of touch receptors, with particular emphasis on their adaptation to certain stimuli [1]. They are employed for interacting with virtual tasks that are usually performed using hands in the real world [2]. Examples include SensAble Technologies' PHANToM, developed by MIT [3], a 3D haptic interface device, and Haptic Technology's CAT® (Computer-Assisted-Touch) family of primarily 2D devices .

Haptic Display means a graphical rendering with the object's haptic properties (e.g. material properties) represented along with it's visualization.

These have been used in special-purpose applications such as in a real-time surgery simulation system in medicine [4]. *Faraz* and *Payandeh* [5] also developed an endoscopic haptic device for applications in medical training and remote medicine.

This paper describes a general practical method for haptic rendering of objects in virtual environments. For the sake of simplification, we restrict the results of the paper to static environments composed of rigid objects. The paper is organized as follows:

Section 2 presents the graphics object modeling data conversion. This includes a practical method for modeling 3D objects and for conversion to a hierarchical data set to construct collision detection and force profiling models. The output form CAD tools is used for rendering and integrated with fast collision detection.

Section 3 discusses collision detection. We include a brief survey of collision detection algorithms and the reasons for adopting RAPID, a fast multi-object collision detection method.

Section 4 describes collision reconstruction and our contact detection algorithm. We contribute an algorithm for detecting accurate contact collision points, edges and faces on the colliding objects and a reconstruction method for rigid object collision.

Finally, section 5 presents experiments and results.

The proposed technique can be used for simulating haptic interactions between a tool and 2D or 3D objects. The graphical models are tessellated with triangular elements. The hardware components of our set-up include a PC running a general graphical software package to display the graphical model of the 3D virtual environment, and normal input devices (keyboard, mouse) to convey to the user a sense of moving and feel of virtual objects (see fig. 1) in this environment.

2. OBJECT MODELING

Our objective is to use high level design tools to model the environment and high level graphics tool to visualize the environment. So we propose a fast collision detection technique between 3D objects as a part of haptic rendering.

This subsystem utilizes exact contact localization method and reconstruction algorithm for contact detection. We display a graphical model of the 3D polygonal objects and update their location as the user manipulates the input devices, detect any collisions between the objects and give exact collision positions. The proposed method is more suitable for the simulating non-convex/convex object collision detection than earlier techniques which are only suitable for convex objects [6].

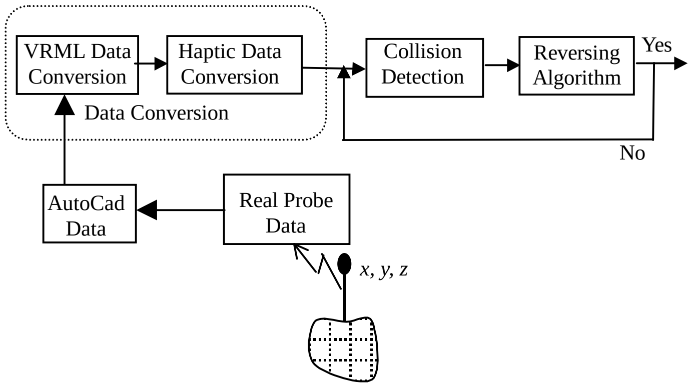

Figure 1: Overview of modeling and collision detection system

For ease of model construction, we chose a widely available CAD system (AutoCAD), and for flexibility and control of the visual display, we chose to use OpenInventor [7]. However, OpenInventor cannot read Autocad's dxf file format. We thus needed to use a conversion utility to generate a readable file format (VRML or OpenInventor file format .wrl, .iv) from the dxf file.

The data structure requirements of most current collision detection algorithms are not compatible with the current graphical data file's structure. Further, the structures used in these algorithms often presents problems for generating force profiles; thus we devised a more appropriate data structure to better support collision detection and evaluation of force profiles.

Current 3D graphic data structures (e.g. Open Inventor, VRML etc.) is an array of triangle structures, each of which contains of three indices into an array of vertices. Disadvantages of this data structure for haptic rendering applications include

a) Complex interface to the collision detection algorithm: A conversion process is needed between the 3D data and the collision detection algorithm.

b) Unable to support material properties. The material property data structure is based on element plates. However, the 3D geometric graphic data structure is based on vertices. The element plate data must be constructed via mapping index parameters. As a result, a complex and robust algorithm will be needed to convert the data structure used in the applications.

To meet this need for a better data structure for haptic rendering applications, we developed a new hierarchical data structure, called a Haptic Data Structure. This data structure, illustrated in fig. 2, supports the collision detection algorithm, material properties and deformation requirements.

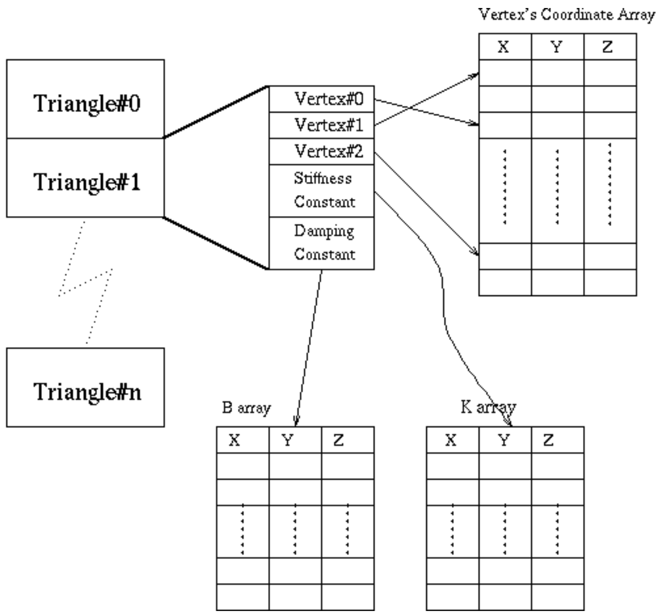

Figure2: The data structure for the 3D Geometric and material property data

Other advantages of the haptic data structure are:

a) Simple interface to the collision detection algorithm.
b) Supports dynamic modification of the object's geometry information.
c) Supports material properties.

3. A COLLISION DETECTION ALGORITHM

Collision detection is a critical component of haptic rendering and a number such algorithms have been developed. Each has its own limitations and advantages with respect to collision detection performance. A brief discussion of collision detection algorithms will help motivate algorithm selection criteria.

A good survey of collision detection algorithms has been done by *Lin* and *Gottschalk* [9]. Relative to our work, a brief survey categorized with respect to polyhedra and volume modeling is as follows:

Type I) Convex planar/curve polyhedra: *Minkowski* difference and convex optimization techniques are used [10] to compute the distance between convex polyhedra by finding the closest points. Geometric Coherence has been exploited to design algorithms for convex polyhedra based on local features [11][12][13] in applications involving rigid motion. Other structured polygon algorithms include cone-tree, k-d tree and octrees [14], 4-D testing, spatial partitioning based space-time bounds or local overlap region testing for swept solid object [15].

More recent work seems to have focused on tighter-fitting bounding volumes. RAPID [6] is a fast algorithm for interference detection based on oriented bounding boxes which approximate geometry better than do axis-aligned bounding boxes. *Barequet*, etal. [16] have also developed a collision detection algorithm based on using oriented bounding boxes for computing hierarchical representations of surfaces for performing collision detection.

Type II) Polygon soups[1] with complex intertwined objects (i.e. multiple objects colliding separately): Only RAPID can efficiently and robustly provide this type of collision detection. The RAPID algorithm can also deal with non-convex polygonal objects.

Type III) Constructive Solid Geometry(CSG): Efficient, accurate and robust computation of bounding volumes remains a hard problem for CSG models described using curved primitives [17][18]. *Cameron* [19] introduced "S-bounds" as a means of speeding up intersection evaluation which is sufficient to determine whether or not the intersection is empty. Based on space partitioning and bounding boxes, *M.A.Ganter* and *B.P.Isaraukura* [15] introduce an S-bounds algorithm whose key technique is to reduce collision detection test points/faces by subdividing the space containing a given object into a set of partitions.

Type IV) Volume Rendering model: In contrast to the surface-based graphical formats, collision detection for voxel-based objects is conceptually simple and does not require any mathematical analysis.

Collisions are detected automatically when a voxel address from one object tries to write into an occupancy map cell that is already occupied by the voxel address of another object [20]. The volume rendering model's collision detection algorithm is simple, but requires a large memory and special hardware acceleration.

Before choosing a particular collision detection algorithm, the collision detection's preliminary problem must be understood clearly: i.e. the simulation environment is regarded as a key point for seeking an efficient algorithm.

It is well known that different simulation environments lead to different collision detection algorithm choices. For practical environments, the basic requirements of the environment are:

1. Planar polygonal approximation of the objects.
2. The objects may be very complex shape (e.g. intertwined objects).
3. Non-convex objects must also be considered.

Based on the collision detection algorithm survey, a collision detection algorithm which is practical for collision detection in a haptic rendering system is the RAPID algorithm.

[1] Polygon soups mean the models contain no adjacency information and obey no topological constraints.

RAPID is a robust and accurate polygon interference detection library for pairs of unstructured polygonal models. It is most suitable for close proximity configurations between smooth surfaces [6]. RAPID is only one algorithm for non-convex objects and can solve multiple object collision problems.

4. CONTACT LOCALIZATION AND RECONSTRUCTION ALGORITHM

Two things must be considered for detecting the point(s) of contact in collision detection simulation:

- the first contact points, edges or surfaces
- the object's trajectory and velocity.

However, because the motion is a sequence of discrete steps, collisions occur under three cases:(fig. 3):

a) The objects interpenetrate.
b) Their boundaries touch.
c) One object passes completely through another.

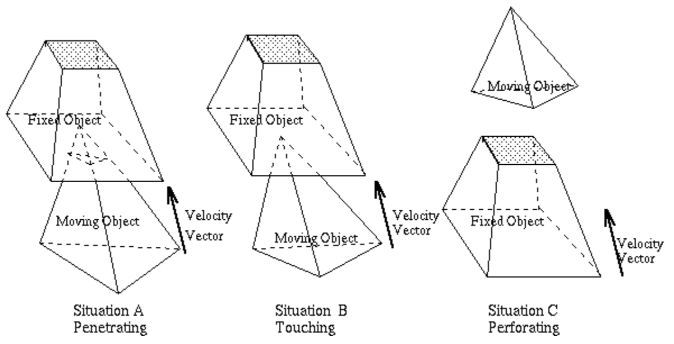

Figure 3: Three situations for object's collision

For case c), the collision detection algorithm cannot produce the correct collision result. The solution here is to reduce the step length to be less than the bounding box's minimum edge length for all objects. For cases a) and b), a new algorithm will be developed which:

a) Finds whether the objects penetrate or touch.
b) Finds the correct reversing vector if they penetrate.
c) Finds the first contact points, lines and polygons.

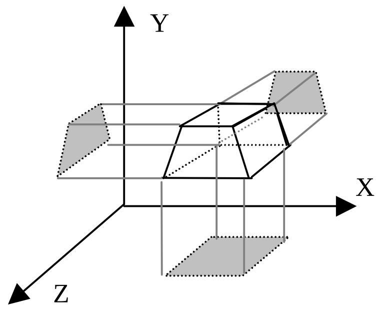

Figure 4: Projections for a 3D object to 2d planes.

To simplify the algorithm complexity for 3D space, a 2D projection method is considered. This method is stable and simple. Using this method, 3D polyhedra are projected onto the three 2D planes (fig. 4). A 3D vector can be decomposed into the three 2D vectors respectively in x-y, y-z, x-z planes (the three 2D vectors can be correspondingly composed into a single 3D vector).

The x-y plane projection is used to illustrate the contact and reconstruction algorithm (fig 5).

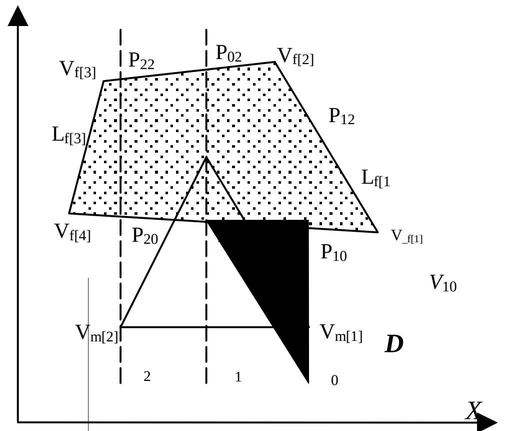

Figure 5: The collision projection on 2D plane

Definitions:

- $\boldsymbol{V}_{f[i]}$ and $\boldsymbol{V}_{m[i]}$ represent the *i*th vertex on the fixed and moving objects respectively;
- $L_{m[j]}$ and $L_{f[j]}$ represent the *j*th line segment on the moving and fixed objects, respectively;
- $l_{m[i]}$ and $l_{f[j]}$ represent the *i*th (*j*th) line through vertex $\boldsymbol{V}_{m[i]}$ ($\boldsymbol{V}_{f[i]}$) where the line's direction is the moving object's velocity vector $\boldsymbol{D}$.

First consider the relation between the vertex $\boldsymbol{V}_{m[i]}$ and the Line segment $L_{f[j]}$. $\boldsymbol{P}_{ij}$ is the intersection point for the line $l_{m[i]}$ and the Line segment $L_{f[j]}$. Then test if point $\boldsymbol{P}_{ij}$ is inside the line segment $L_{f[j]}$. If not, the point $\boldsymbol{P}_{ij}$ will be ignored.

Second find the penetrating vectors $\boldsymbol{V}_{ij}$ where
$\boldsymbol{V}_{ij} = (\boldsymbol{P}_{ij}. - \boldsymbol{V}_{m[i]})$.

Third find the maximum penetrating distance and reversing vector. The penetrating distance values d_{ij} are given by
$d_{ij} = \boldsymbol{V}_{ij} \bullet \boldsymbol{D}$. If $d_{ij} \geq 0$, the corresponding vertex $\boldsymbol{V}_{m[i]}$ penetrates the fixed polyhedron. From the set of d_{ij}'s, a maximum value can be found. The maximum value corresponds to at least one vertex. corresponding array of vertices $\boldsymbol{V}_{m[i]}$'s (with the same maximum *d*) is regarded as the set of contact points. If neighboring contact points are independent, then they are the touch vertices. If neighboring contact points are vertices for the same line segment of a polyhedron, an edge touches. If neighboring contact points form a polygon, a face (polygon) contact is said to occur.

The complete contact problem is now solved, except for certain singular situations as shown in fig. 6. When the moving object's colliding vertex/vertices is/are not inside the fixed object, the singular situation would happen.

Figure 6: A sigular situation for collision based 2D plane

However, a similar procedure can solve this singular problem. This procedure is very similar to the above, except the vertex is changed from the moving object's $\boldsymbol{V}_{m[i]}$ to the fixed object's $\boldsymbol{V}_{f[i]}$, the Line segments changed from the fixed object's $L_{f[j]}$ to the moving object's $L_{m[j]}$, corresponding the lines for the moving object $l_{m[i]}$ also changed to the fixed object's $l_{f[j]}$. Finally, the direction vector $\boldsymbol{D}$ must be reversed and the new direction vector is -$\boldsymbol{D}$. Detailed pseudo-code is shown in the appendix.

5. EXPERIMENT SETUP AND RESULTS

This implementation uses a fixed step length technique, a fast collision algorithm (RAPID) and the collision response-based reconstruction algorithm. Fig. 7 shows an overview of the algorithm.

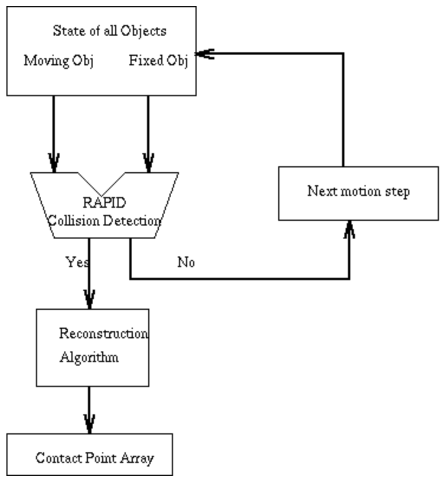

Figure 7: Algorithm overview.

Multiple collisions within one step can be processed as serial collisions in one simulation step. The algorithm solves not only simultaneous collisions as a series of individual collisions, but also is suited for non-convex shapes objects. It updates the state of the objects immediately after a collision.

A test environment containing user-defined virtual objects has been built. An environment containing fixed (static) objects and moving (dynamic) objects has been tested. Users can build and edit test object models using commercial 3D CAD software such as AutoCad. During the experiment, the user can manipulate (move/rotate) the objects using the mouse and keyboard and see virtual collisions on the screen.

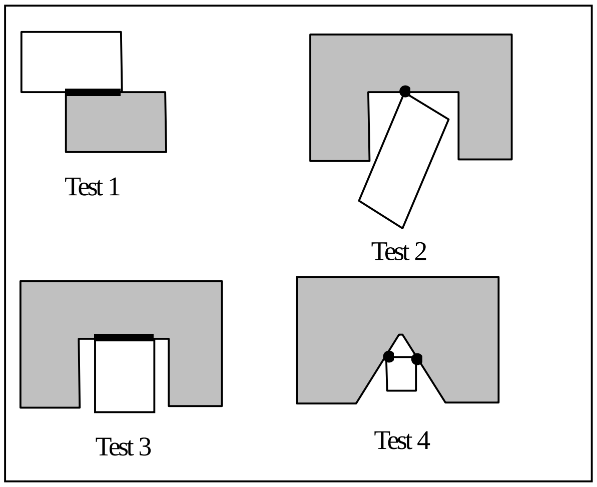

Figure 8. Experimental test cases.

Fig. 8 shows four test cases of varying difficulty ranging from a simple one-point collision of two convex objects through a line collision to multiple point collision of non-convex objects. The contact points, edges and faces are highlighted in the figure (and on the screen). Detailed contact coordinate information is displayed to the user on the screen, along with the reversing vector. The algorithm's performance is shown in Table 1. The experiment was run on an NT workstation with a Pentium II-333 CPU and 128MB of memory.

Table 1. Performance of reconstruction algorithm.

Example No.	#1	#2	#3	#4
# of colliding lines	32	54	409	1062
# of colliding vertices	22	31	205	502
Running Times(sec)	.00015	.00031	.991	5.00

6. CONCLUSIONS

This paper proposes a practical approach for efficient collision detection, object reconstruction for the rigid object collision. The proposed approach enables us to simulate complex non-convex objects (even for intertwined shapes) free moving and colliding in 3D space in real time. As our contribution, a realistic animation of rigid object collision can be simulated via the RAPID collision detection algorithm and the reconstruction algorithm. In additional, a practical data structure for graphical and force data is introduced in this paper. Using this data structure, a practical and efficient interface is built between the algorithm and the 3D graphical modeling data.

For future research, we plan to improve RAPID algorithm for working with the deformable object collision detection. We will also try to combine the approach with deformation analysis such as spring-matrix elastic model [21] or thin plate deformation analysis [22] with fast collision detection and reconstruction algorithm combine with real data measurements.

REFERENCES

[1] Burdea,C.G., D.Gomez, N.Langrana, E.Roskos, and P.Richard,"Virtual Reality Graphics Simulation with Force Feedback" *International Journal in Computer Simulation*, ABLEX Publishing, Vol. 5, pp.287-303. 1995

[2] C.Basdogan, C.H.Ho, and M.A.Srrinivasan, "A Ray-Based Haptic Rendering Technique for Displaying Shape and Texture of 3D Objects in Virtual Environments" *Proceedings of the Dynamic System And Control Division*, DSC-Vol.61, ASME, pp.77-84. 1997.

[3] Massie,T. and K.Salisbury, "The PHANToM Haptic Interface: A Device for Probing Virtual Objects," *ASME Winter Annual Meeting*, DSC-Vol.55-1, ASME, New York, pp.295-300. 1994.

[4] S.Cotin, H.Delingette, "Real-time Surgery Simulation with Haptic Feedback using Finite Elements" *International Conference on Robotics & Automation, Proceedings of the 1998 IEEE* pp.3739-3744. 1998.

[5] A.Faraz and S.Payandeh, "Design and Analysis of Tunable Springs in Haptic Interface of Endoscopic Graspers," *Proceedings of the Dynamic System And Control Division*, DSC-Vol.61, ASME, pp.69-76. 1997.

[6] S. Gottschalk, M. Lin, and D. Manocha. "Obb-tree: A hierarchical structure for rapid interference detection" *Proc.. of ACM Siggraph'96*,pp.171-180, 1996.

[7] Josie. Wernecke, et al "The Open Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor, Release 12" *Addison-Wesley Developers Press* ISBN 0-201-62495-8. 1994.

[8] Keith, Rule. "3D Graphic File Formats: A Programmers Reference" *Addison-Wesley Developers Press,*
ISBN 0-201-48835-3. 1996.

[9] M. Lin, and S. Gottschalk. "Collision detection between geometric models: a survey" *Proceedings. of IMA Conference on Mathematics of Surfaces 1998*.

[10] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. "A fast procedure for computing the distance between objects in three-

dimensional space". *IEEE J. Robotics and Automation, vol. 4* pp.193-203, 1988.

[11] D. Bara. "Curved surfaces and coherence for non-penetrating rigid body simulation". *ACM Computer Graphics, 24(4)* pp.19-28, 1990.

[12] M.C. Lin and John F. Canny. "Efficient algorithms for incremental distance computation". *In IEEE Conference on Robotics and Automation*, pp.1008-1014, 1991.

[13] M.C. Lin. "Efficient Collision Detection for Animation and Robotics". *PhD thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley*, December 1993.

[14] H.Samet. "Spatial Data Structures: Quadtrees, Octrees and Other Hierarchical Methods." *Addison Wesley*, 1989.

[15] M.A.Ganter, B.P.Isarankura " Dynamic Collision Detection Using Space Partitioning" *In transactions of the ASME, 150/Vol.115*, March 1993.

[16] G. Barequet, B. Chazelle, L. Guibas, J. Mitchell, and A. Tal. "Box-tree: A hierarchical representation of surfaces in 3d." *In Proc. of Eurographics'96*, 1996.

[17] C.M. Hofmann. "Geometric and Solid Modeling." *Morgan Kaufmann*, San Mateo, California, 1989.

[18] J. Keyser, S. Krishnan, and D. Manocha. "Efficient and accurate brep generation of low degree sculptured solids using exact arithmetic." *In ACM/SIGGRAPH Symposium on Solid Modeling*, pp.42-55, 1997.

[19] S. Cameron. "Approximation hierarchies and s-bounds". *In Proceedings. Symposium on Solid Modeling Foundations and CAD/CAM Applications*, pp.129-137, Austin, TX, 1991.

[20] S F.Frisken Gibson. "Beyond Volume Rendering: Visualization, Haptic Exploration, and Physical Modeling of Voxel-based Objets" *Mitsubishi Electric Research Laboratories, Cambridge Research Center, Technical Report* 95-04.

[21] Ugur. Gudukbay, Bulent. Ozguc, Yilmaz. Tokad. "A Spring Force Formulation For Elastically Deformable Models" *Computer And Graphics, Vol.21 No.3* pp.335-346, 1997.

[22] Hong Qin, D. Terzopoulos. "D-Nurbs: A Physics-Based Framework for Geometric Design" *Visualization and Computer Graphics, Vol.2, No.1*, pp.85-96, March 1996.

APPENDIX

PSEUDO-CODE FOR RECONSTRUCTION ALGORITHM

```
Initialize $\mathbf{V}_{f[i]}$, $\mathbf{V}_{m[i]}$, $L_{m[j]}$, $L_{f[j]}$. Set velocity vector $\mathbf{D}$ for moving
object and initialize $l_{m[i]}$ and $l_{f[j]}$. Initialize maximum distance
$d_{max}$.
// calculate $d_{max}$ for the fixed object:
$\mathbf{P}_{ij}$ =Intersection ($L_{f[j]}$ , $l_{m[i]}$) ;
If $\mathbf{P}_{ij}$ inside Line segment $L_{f[j]}$, keep the $\mathbf{P}_{ij}$ ;
    Else $\mathbf{P}_{ij}$ = NULL;
$\mathbf{V}_{ij} = (\mathbf{P}_{ij} - \mathbf{V}_{m[i]})$ when Pij = Null;
for (m=0; m ≤ i; m++) \{
   for (n=0; n ≤ j; n++)
   {   if $d_{ij} = \mathbf{V}_{ij} \bullet \mathbf{D} \geq 0$.
      {   if $d_{ij} \geq d_{max}$ {
            {$d_{maxf} = \mathbf{V}_{ij} \bullet \mathbf{D}$ ;
              Add $\mathbf{P}_{ij}$ I to contact array;
              Set reversingVector $\mathbf{V}_{ij}$ r;
            }
      }
   }
}
// calculate $d_{max}$ for the moving object ($d_{maxm}$ ) similarly
$d_{max}$ =max($d_{maxf}$ , $d_{maxm}$)
```