Haptic Interaction in Virtual Environments

Diego C. Ruspini[1], Krasimir Kolarov[2] and Oussama Khatib[1]

Robotics Laboratory[1]
Computer Science Department
Stanford University, Stanford, CA 94305

Interval Research Corporation[2]
1801 Page Mill Road, Building C, Palo Alto, CA 94304

Abstract

We present a haptic rendering framework that allows the tactile display of complex virtual environments. This framework allows surface constraints, surface shading, friction, texture and other effects to be modeled solely by updating the position of a representative object, the "virtual proxy." This abstraction reduces the task of the haptic servo control loop to the minimization of the error between user's position and that of the proxy. This framework has been implemented in a system that is able to haptically render virtual environments of a complexity that is near and often in excess of the capabilities of current interactive graphic systems.

1 Introduction

Haptic rendering systems simulate the forces generated by objects, allowing a person to sense and interact with a virtual environment through touch. Coupled with visual feedback these systems can enhance a users ability to perform tasks [6,14,18] as well as add physical realism to interactive graphic systems.

Traditional systems apply forces proportional to the amount of penetration into a virtual volume. For simple geometries, like spheres and planes, the direction and amount of penetration is easy to determine and the simplicity of this approach has allowed it to be used to study many interesting simple environments. These *penalty-based* methods, however, have a number of drawbacks. When contact exists with multiple surfaces it is often difficult to determine the nearest exterior position. In the worst case, a global search of all the primitives may be required to find the nearest exterior surface. If the forces from multiple objects are added the resultant force can create unstable motions that could cause device damage or user injury. In addition, graphics models, which most often consist of infinitely thin polygons, lines and points, do not contain sufficient internal volume to generate the repulsion forces needed to prevent the probe form passing through the object.

To address the limitations of penalty-based approaches we have developed a new *constraint-based* control framework built around the notion of the "virtual proxy." The virtual proxy is a representative object that substitutes for the physical finger or probe in the virtual environment. The movement of the proxy towards the goal (user's finger location) bares a strong similarity to reactive path planning in robotics applications. When unobstructed, the proxy moves directly towards the goal. When the proxy encounters an obstacle, direct motion is not possible, but the proxy may still be able to reduce the distance to the goal by moving along one or more of the constraint surfaces. The motion is chosen to locally minimize the distance to the goal. When the proxy is unable to decrease its distance to the goal, it stops at the local minimum configuration.

Constraint-based methods were first proposed for haptic applications by Zilles and Salisbury [25]. A point (god-point) was constrained by the objects in the environment. A topology of the rendered object was required to prevent the god-point from falling through small gaps commonly found in graphical models. This computationally expensive preprocessing step limited the interactivity of the system.

Our exploration has focused on creating a system that is capable of rendering virtual environments common in graphic applications. These environments are typically represented by a large number (20,000 or greater) of unordered polygonal surfaces, lines and points (polygon soups). In our implementation no topological information is assumed and intersecting polygons are permitted. In addition, surface normals for shading and image mapped textures, which are often available in graphic applications, are used in our system to enhance the richness and complexity to the haptic scene.

2 Virtual Proxy Update

For simplicity we represent the virtual proxy as a massless sphere that moves among the objects in the environment. Because of small numerical errors, polygons that are intended to share a common edge often contain gaps. The radius of the proxy is made large enough to avoid falling through the holes in the underlying model. We will also assume that all the obstacles in the environment can be divided into a finite set of convex components.

Haptic devices require high controller servo rates – typically over 1000Hz– to achieve stability and high disturbance rejection. During each servo tick, a goal configuration for the proxy is found and an attempt is made to move the proxy to this configuration by direct linear motion. The goal configuration is initially taken to be at the end point of the haptic device. This position, however, will change as the proxy encounters obstacles in the environment.

The volume swept by the virtual proxy, as it moves during a given time period, is checked to see if it penetrates any primitive in the environment. Because the path of the proxy is linear, this test involves determining whether a line-segment, specified by the proxy's current and goal configurations, falls within one proxy radius of any object in the environment.

If the proxy's path does not collide with any obstacles, the proxy is allowed to move directly towards the goal. If one or more interfering primitives are found, the proxy's position is advanced until it makes contact with the first obstacle in its path. To model this interaction efficiently, we consider the configuration space of the proxy, where the *configuration-space obstacles* (C-obstacles)[12], consist of all points within one proxy radius of original obstacles. In this space, the position of the proxy is identified by a point while all C-obstacles have continuously defined surfaces and non-zero thickness. A unique constraint plane can be found where the line segment representing the proxy's path intersects the C-obstacle. An example of configuration space, C-obstacles and proxy constraint planes is shown in Figure 1.

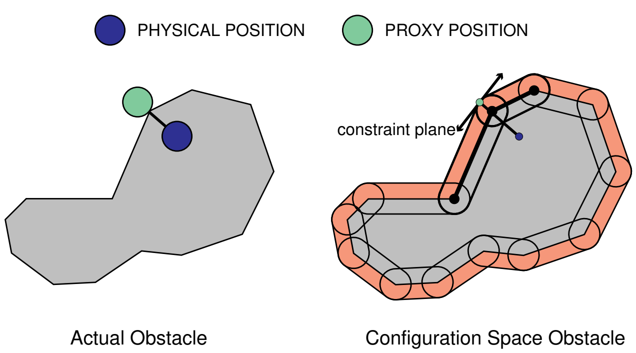

Figure 1: Configuration Space Obstacles & Constraint Planes

Each constraint plane limits the directions of motion to the half-space above the plane. The intersection of all such half-spaces defines a convex, unbounded polyhedron. The new sub-goal is a point within this convex region (the local free-space) that is closest to the user's position. All the constraint planes go through the current proxy position, therefore by translating the current proxy position to the origin the problem can be written compactly as

$$\begin{aligned} &\text{minimize } \|x - p\| \text{ subject to} \\ &\hat{n}_1^T x \geq 0, \\ &\hat{n}_2^T x \geq 0, \\ &\vdots \\ &\hat{n}_m^T x \geq 0. \end{aligned} \tag{1}$$

where p is the vector from the current proxy position to the user's position, $\hat{n}_i$, $0 \leq i \leq m$, are the unit normals of the constraint planes, and x is displacement from current proxy position to the new goal configuration. Once x is found the iteration can continue.

This problem may be solved using a general quadratic programming package like Gill et. al [8]. There are, however, many simplifications that make a simpler and faster solution possible. In our implementation, this problem is solved in two stages. The minimum set of active constraint planes is identified first; this set is then used to find a new sub-goal position.

The convex free-space polyhedron has a dual space consisting of the convex hull defined by the points, $-\hat{n}_i$, $0 \leq i \leq m$ (the outward normals of the planes forming the free-space region) and the origin (plane at infinity). The constraint planes that bound the solution can be found by determining the closest face, edge, or vertex of this hull to a point $\hat{p}$, a unit vector with the same direction as p. This problem may be solved using the same algorithm employed in the collision detection process[7]. The vertices of the closest face, edge or vertex indicate which of the corresponding constraint planes bound the solution.

Once the bounding planes have been determined Equation 1 may be solved using only the identified planes as constraints. With the inequalities replaced by equalities the problem can be solved easily using Lagrange multipliers as is described by Zilles in [25]. The solution x represents the displacement from the current to the new sub-goal. Since, at most, three planes can be active at one time, the entire solution can be found in $O(m)$ time, where m is the original number of constraint planes.

Each iteration monotonically decreases the distance between the proxy and the user's position, and thus ensures that the movement of the proxy will be stable if the input from the user is stable.

3 Force Shading

Most graphic interfaces permit the specification of surface normals on the vertices of polygonal surfaces. This information is used to alter the lighting model on the surface to give it the appearance of being smooth[9,19]. Morgenbesser and Srinivasan [17] were the first to demonstrate that a similar haptic effect could be created. Their solution changes the direction of the normal force while retaining the magnitude caused by the penetration of the original surface. While this technique produces compelling shading effects it is unclear how to extend this approach to handle multiple intersecting shaded surfaces or support additional surface effects, such as friction or texture.

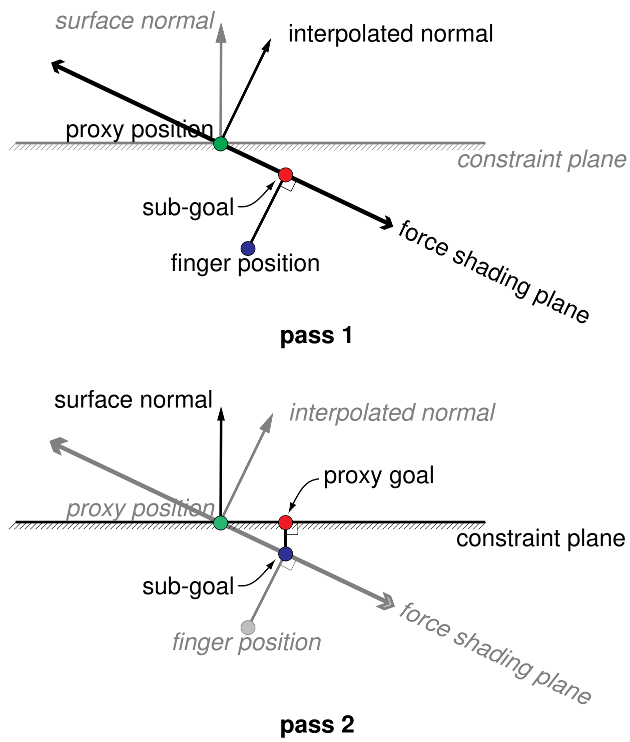

Figure 2: Two Pass Force Shading with User Supplied Normals

When a polygonal surface is encountered for which user-specified normals exist, a new local surface normal is calculated by interpolating the normals from the vertices of the polygon. This interpolation is very similar to that required for Phong shading in computer graphics applications[19]. The interpolated normal specifies a new constraint plane going through the contact point. The algorithm proceeds by first finding a new sub-goal using the interpolated planes instead of the original constraint planes. This sub-goal is then treated as the user's finger position and a second pass of the update procedure is performed to obtain the final sub-goal configuration for this iteration. This second pass is performed using the true (non-interpolated) constraint planes. This process considers the effect of all constraint surfaces in both passes and produces the correct result even if multiple force shaded surfaces exist. This process is illustrated in Figure 2.

If after the first pass the sub-goal configuration is above all the true constraint planes, the sub-goal is first projected back onto the nearest true constraint plane. This ensures that new sub-goal point will always lie on the object surface so that surface effects like friction and texture will be handled correctly.

Force shading may increase the distance between the user's finger position and the position of the proxy. This increase implies that the surface is active and can add energy to the haptic/user system. This added energy, however, is typically quite small because the difference between the interpolated and true constraint planes is typically quite small.

The difference between the force shaded surface and a normal surface is illustrated in Figures 3 and 4. In both figures the difference between the actual user position and the position of the virtual proxy are shown as the user's finger follows a circular path around a ten-sided polygonal approximation of a circular object. For compactness, only the first quadrant of the circle is shown. Because the constraint surface is curved, the differences between the position of the proxy and the finger are best illustrated in polar coordinates. The angular displacement corresponds roughly to the amount to which the user's finger is pulled tangentially as the user moves around the cylinder. The radial displacement indicates roughly the amount of force that the user feels pushing against the surface of the object.

As seen in Figure 3, a strong discontinuity occurs when the proxy finally reaches the edge of the object. In Figure 4, surface normals have been specified on the vertices. The resulting movement of the proxy shows that no tangential force is felt by the user as the finger is moved around the object. This is what would be expected if the user were moving around a perfectly circular object. In the radial plot it can be seen that the proxy still follows the underlying surface of the object while the motion of the proxy is continuous and therefore feels smoother to the user.

4 Friction

Surface effects can also be created solely by altering the movement of the proxy. Several researchers have proposed methods to simulate static, dynamic and viscous friction [2,4,14,16,23].

Static friction is particularly simple to model within the virtual proxy framework. The force exerted on the proxy by the user can be estimated by the equation $f = k_p(p - v)$, where p is the position of the proxy, v is the position of the finger and k_p is the proportional gain of the haptic controller. For a given constraint plane let f_n and f_t be the components of the force on the proxy normal and tangential to the constraint plane, respectively. If the given constraint surface has a static friction parameter μ_s, then the proxy is in static contact if $\|f_t\| \le \mu_s \|f_n\|$. This is equivalent to stating that the user's finger position lies in the friction cone of the surface. When any constraint surface is in static contact the proxy's position is left unchanged.

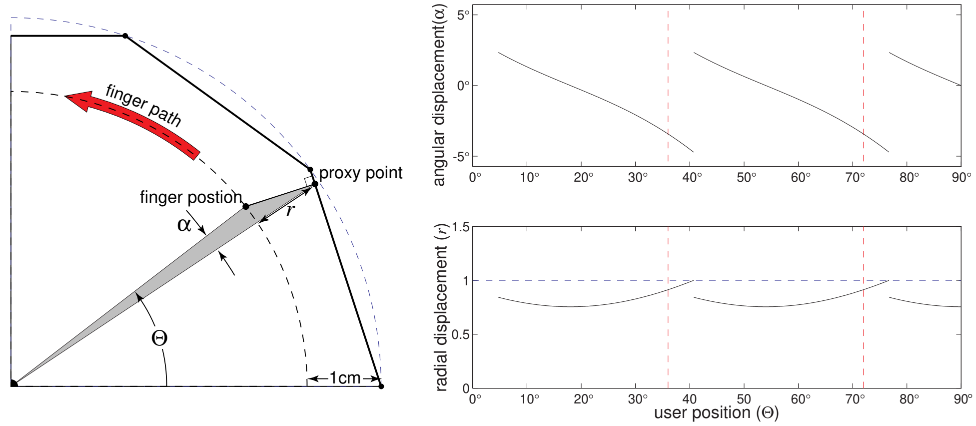

Figure 3: Virtual Proxy Path without Force Shading

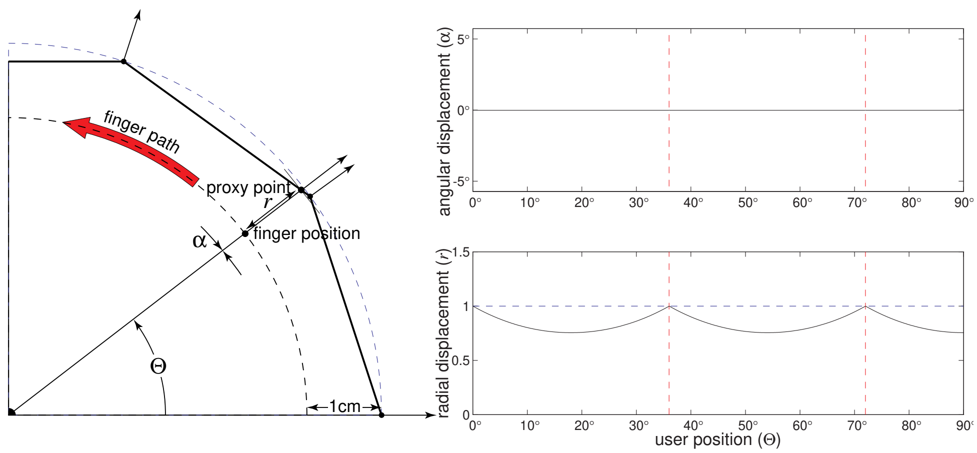

Figure 4: Proxy Path with Force Shading

Viscous and dynamic friction can be modeled by observing the motion of a one dimensional object. The equation of motion of an object with mass m moving in a viscous field, along a surface that exhibits dynamic friction is

$$f - \mu_d f_n = m\ddot{x} + b\dot{x}, \qquad (2)$$

where b is a viscous damping term, μ_d is the coefficient of Coulomb friction. As the mass of the object approaches zero, the body quickly reaches its saturation velocity. In dynamic equilibrium, the velocity of the object is given by

$$\dot{x} = \frac{f - \mu_d f_n}{b}. \qquad (3)$$

This limit can be used to bound the amount that the proxy can travel in a given clock period. When multiple constraint surfaces exist, the lowest velocity bound is taken as the limit of the proxy's movement. In the event that the maximum velocity is negative, then the dynamic friction term is sufficient to resist all movement and the proxy's position is not changed. If $b = 0$ no viscous term exists and the maximum velocity is not limited. This approach does not use the finger's velocity and is therefore not susceptible to errors caused from trying to estimate this value from a finite number of encoder readings.

5 Texture

Haptic textures were first demonstrated by Minsky et al.[16] for the haptic display of height fields on a two degree of freedom planar haptic display. In our system a image-based texture map can be used to modulate any of the surface parameters—friction, viscosity or stiffness—to create different haptic effects. At present, the texture values are only evaluated at the proxy position at the beginning of the each clock cycle. This produces a convincing effect with slowly changing textures. In reality, the texture values should be evaluated as the proxy moves along the surface of the object, so as to assure that a significant change is not missed as the proxy travels on the surface.

Another interesting approach to the modeling of textures is based on the modification of the force-shading constraint planes in a manner similar to that employed in bump mapping in computer graphics[3]. In our current system a texture bump-map generates at most one additional constraint plane for each textured surface. This approach is adequate to model continuously differentiable textured surfaces. Grooved or cratered surfaces which may contain multiple simultaneous constraints require additional constraint planes and the path of the proxy in the texture space must be followed to ensure that the proxy would not become constrained by another texture feature as it moves along the surface.

6 System Implementation

The system we have developed runs on two processors: the haptic server and the application client. The separation of the haptic and application/graphic processes was first proposed by Adachi et al. [2]. Decoupling the low-level force servo loop from the high-level control is needed since the haptic servo loop must run at a very high servo rate, typically greater than 1000Hz. Most graphic application programs typically run at a much slower rate (around 30Hz).

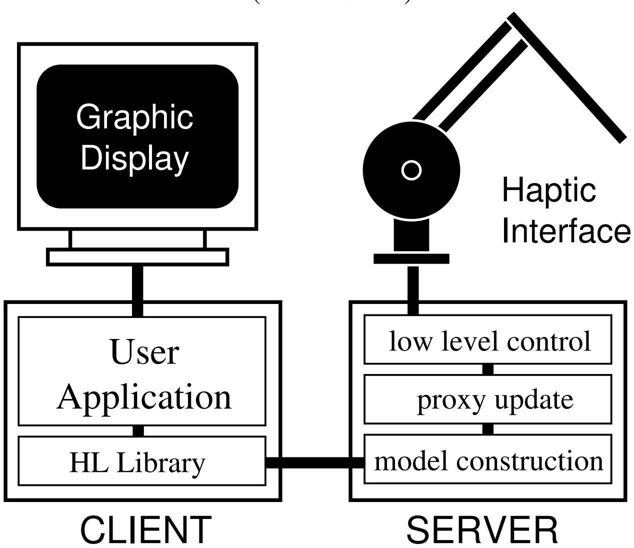

Figure 5: System Architecture

In our system, illustrated in Figure 5, the bulk of haptic rendering effort is placed on the haptic server, thus freeing the client machine to perform the tasks required by the user's application. The haptic server receives high level commands from the client, tracks the position of the haptic device, updates the position of the virtual proxy, and sends control commands to the haptic device. This arrangement places the performance bottle-neck on the haptic server CPU rather than on the I/O channel, an advantage given current technological trends.

6.1 The Client Application

Applications communicate to the haptic server through the HL network interface library, we have developed. The current library supports a limited set of the functions provided by the GL graphics library. The HL Library allows users to define objects as a collection of primitive objects — points, line segments, or polygons. Transformations are provided to allow objects and primitives to be freely translated or rotated. Surface normals and texture coordinates can be associated with polygon's vertices to allow the definition of force shaded or textured surfaces. Object hierarchies and material properties such as friction and stiffness may also be defined.

6.2 Model Construction

Once the modeling commands are received from the client, they must be stored in a form suitable for haptic rendering. Vertices are transformed into local object frames and meshes and line segment sequences are represented as sets of independent convex bodies.

Because each object is normally constructed from a large number of primitives, a naive test based on checking if each primitive is in the path of the proxy would be prohibitively expensive. In general the proxy's path will be in contact at most a small fraction of the underlying primitives. In our approach a hierarchical bounding representation for the object is constructed to take advantage of the spatial coherence inherent in most objects. The bounding representation, based on spheres, is similar to that proposed by Quinlan[20].

6.3 Low Level Control

A virtual proxy reduces the task of the low-level servo controller to minimization of the error between the configuration of the proxy and position of the haptic device. In our current implementation a simple operational space proportional derivative (PD) controller is used[11].

The low-level control loop may be separated from the contact/proxy update loop to guarantee stability of the system even in the presence of a large number of objects. By running the control loop at a high fixed clock rate, stability can be ensured and the fidelity of the haptic display made to degrade gracefully as the complexity of the environment is increased. If the proxy update procedure is unable to maintain the same rate as the controller, objects feel "sticky." While this effect may not be desirable, it is preferable to permitting unstable and dangerous behavior of the haptic device.

7 Results

Our haptic library has been successfully tested on a large number of polygonal models, including some containing more than 24000 polygonal primitives. In our tests the client computer was a SGI Indigo2 High Impact running IRIX 6.2 while the haptic server was a 200Mhz Pentium Pro running Linux 2.0.2. Communication between computers was made through a standard ethernet TCP/IP connection. The haptic device employed was a ground based 3 dof PHANToM manipulator. The server produced stable results with position gains over 1800 Newton/meter and no artificial damping. Computational expense grows at a logarithmic rate with the number of polygons.

The current system is quite adept in modeling a large number of geometric models. Figure 6 shows a VRML model of an AT-AT from Star Wars containing over 11,000 polygons. Force shading is used to model some of the apparently curved surfaces of the underlying polygonal model. A high servo rate is obtained even in configurations containing over 40 simultaneous contact surfaces between the proxy and the model. The location of the proxy, rather than finger position is displayed to the user, to further enhance the sense of rigidity of the model [24].

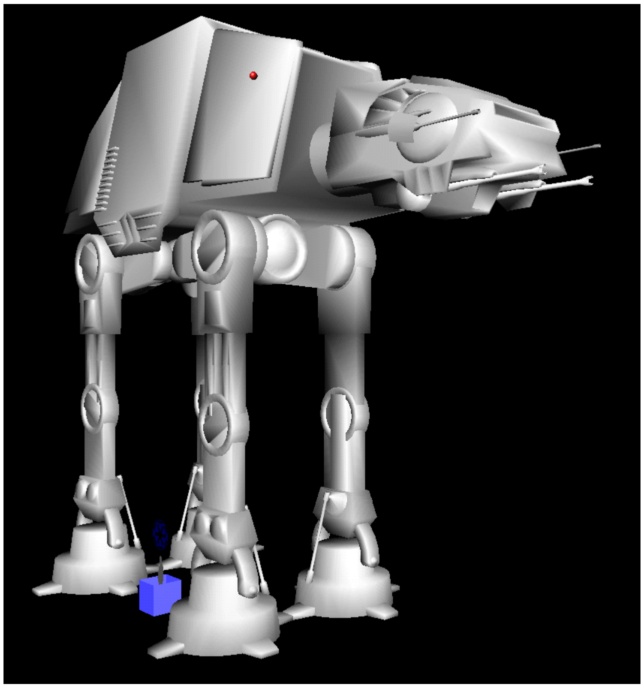

Figure 6: Haptic AT-AT (11088 polygons)

8 Future Work

While the current system is able to model a wide variety of objects and material properties, it only supports limited manipulation of the objects in the environment. As is the case with graphic systems, movement is simulated by re-rendering the moving objects at different locations. These discrete motion steps, which are specified by the client process, result in a discontinuous jerky motion. Furthermore, it is also possible, in some cases for the proxy to lie outside of an object at one time step and within it the next. We are currently looking at several ways of allowing the application program to easily endow haptic objects with smooth, continuous and dynamic motions.

The virtual proxy framework can be expanded to handle other geometric representations such as NURB surfaces and volumetric data sets directly, without transforming these representations into polygonal surface models. This ability is beneficial when the cost of this transformation is prohibitive or can degrade the interactivity of the application.

Acknowledgments

We wish to thank Oliver Brock, Kyong-Sok Chang, Eugene Jhong, Karon MacLean, Robert Shaw and Bill Verplank for their helpful insights and discussion in preparing this paper. The AT-AT by H.H.C. from Avalon Viewpoint archive. The financial support of NASA/JSC, grant NGT 9-6, Boeing, and NSF, grant IRI-9320017, are gratefully acknowledged.

References

[1] Avila, R. S., Sobierajski, "A Haptic Interaction Method for Volume Visualization," *Visualization '96 Proceedings*, Oct. 1996.

[2] Adachi, Y., Kumano, T., Ogino K., "Intermediate Representation for Stiff Virtual Objects." *Proc. IEEE Virtual Reality Annual Intl. Symposium '95,* (March 11-15), pp. 203-210.

[3] Blinn, J., "Simulation of Wrinked Surfaces," SIGGRAPH 78 Proceedings, (August 1978), pp. 286-292.

[4] Buttolo, P., Kung, D., Hannaford, B., "Manipulation in Real, Virtual and Remote Environments." *Proc. IEEE Conference on Systems, Man and Cybernetics* (August 1990), pp. 177-185.

[5] Craig, J., "Introduction to Robotics Mechanics and Control," *Addison-Wesley Pub. Co.,* 1989.

[6] Finch, M., Chi, V., Taylor, R. M. II, Falvo, M., Washburn, S., Superfine, R., "Surface Modification Tools in a Virtual Environment Interface to a Scanning Probe Microscope," *Proc. 1995 Symposium on Interactive 3D Graphics*, pp13-18, April 1995.

[7] Gilbert, E. G., Johnson, D.W., Keerthi, S. S., "A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space," *IEEE J. of Robotics and Automation*, Vol.4, No. 2, April 1988.

[8] Gill, P., Hammarling, S., Murray, W., Saunders, M., Wright, M., "User's Guide to LLSOL," *Stanford University Technical Report SOL 86-1*, (January 1986).

[9] Gouraud, H. "Continuous Shading of Curved Surfaces*." IEEE Transactions on Computers*, C-20(6):pp 623-629, June 1971.

[10] Iwata H., Noma, H., "Volume Haptization*," IEEE 1993 Symposium on Research Frontiers in Virtual Reality*, pp. 16-23, October 1993.

[11]Khatib, O., "A Unified Approach to Motion and Force Control of Robot Manipulators: The Operational Space Formulation," IEEE Journal of Robotics and Automation, Vol 3., No 1., 1987.

[12] Latombe, Jean-Claude, "Robot Motion Planning," Kluwer Academic Publishing, 1991, pp 58-152.

[13] Lin, M., Canny, J. F., "A Fast Algorithm for Incremental Distance Calculation," *International Conference on Robotics and Automation*, pp. 1008-1014, May 1991.

[14] Mark, W. R., Randolph,S. C., Finch M., Van Verth,J. M., Taylor II,R. M., "Adding Force Feedback to Graphics Systems: Issues and Solutions," *SIGGRAPH 96 Proc.*, (Aug. 1996), pp. 447-452.

[15] Massie, T.M., Salisbury, J.K., "The PHANToM Haptic Interface: A Device for Probing Virtual Objects." ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems 1994, In *Dynamic Systems and Control 1994* (Chicago, Illinois, Nov. 6-11), vol. 1, pp.295-301.

[16] Minsky, M. D. R., "Computational Haptics: The Sandpaper System for Synthesizing Texture for a Force-Feedback Display." PhD thesis, MIT, June 1995.

[17] Morgenbesser, H. B., "Force Shading for Haptic Shape Perception in Haptic Virtual Environments." M.Eng. thesis, MIT, Sept. 1995.

[18] Ouh-Young, M., "Force Display in Molecular Docking*," Ph. D. Dissertation, UNC Chapel Hill, UNC-CH CS TR90-004*, Feb., 1990.

[19] Phong, B. T., "Illumination for Computer Generated Pictures." *Communications of the ACM*, 18(6), pp311-317, June 1975.

[20]Quinlan, S., "Efficient Distance Computation between Non-Convex Objects," *Int. Conf. of Rob. and Automation*, (April 1994).

[21] Ruspini, D., Kolarov, K., Khatib, O., "The Haptic Display of Complex Graphical Environments," SIGGRAPH 97 Proceedings, (August 1997).

[22] Salcudean, S. E., Vlaar, T. D., "On the Emulation of Stiff Walls and Static Friction with a Magnetically Levitated Input/Output Device," ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems, Dynamics Systems and Control, pp.123-130, April 1995.

[23] Salisbury, K., Brock, D., Massie, T., Swarup, N., Zilles, C., "Haptic Rendering: Programming Touch Interaction with Virtual Objects," Proc. 1995 Symposium on Interactive 3D Graphics, April 1995, pp. 123-130.

[24] Srinivasan, M. A., Beauregard, G. L., Brock, D. L., "The Impact of Visual Information of the Haptic Peception of Stiffness in Virtual Environments," ASME Winter Annual Meeting, November 1996.

[25] Zilles, C. B., Salisbury, J. K., "A Constraint-based God-object Method for Haptic Display." ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems 1994, *Dynamic Systems and Control 1994* (Chicago, Illinois, Nov. 6-11), vol. 1, pp.146-150.