The Haptic Display of Complex Graphical Environments

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

Stanford University[1]

Interval Research Corporation[2]

Abstract

Force feedback coupled with visual display allows people to interact intuitively with complex virtual environments. For this synergy of haptics and graphics to flourish, however, haptic systems must be capable of modeling environments with the same richness, complexity and interactivity that can be found in existing graphic systems. To help meet this challenge, we have developed a haptic rendering system that allows for the efficient tactile display of graphical information. The system uses a common high-level framework to model contact constraints, surface shading, friction and texture. The multi-level control system also helps ensure that the haptic device will remain stable even as the limits of the renderer's capabilities are reached.

CR Categories and Subject Descriptors: C.3 [Special Purpose and Application-Based Systems]: Real-time Systems; I.3.7 [Three-Dimensional Graphics and Realism]: Virtual Reality; I.3.4 [Graphics Utilities]: Device Drivers; I.6.8 [Simulation and Modeling]: Distributed.

Additional Keywords: haptic, force feedback, force shading, contact constraints, friction model, haptic texture, virtual environments.

1 INTRODUCTION

Haptic devices allow physical interaction with virtual environments, enhancing the ability of their users to perform a variety of complex computer interaction tasks [7,16,20]. Several technological advances are required, however, for haptic systems to achieve the ubiquity of graphic systems.

A successful haptic system must complement existing graphic devices. Current desktop graphic systems are capable of rendering over 20,000 shaded and textured polygonal surfaces at interactive (30Hz) rates. In comparison, most of the current haptic systems are only capable of representing a few dozen geometric primitives.

Because visual and tactile tasks are often closely intertwined, the haptic system should be capable of representing the surfaces and objects that are commonly found in graphic environments. These environments are usually composed of many zero-width polygons, lines and points. The haptic system should also make use of additional graphical information such as surface normals and texture maps which add to the complexity and richness of graphical environments.

Furthermore, graphic models often contain intersecting surfaces and gaps between primitives, and the topology of the model is seldom known. The haptic system should avoid, as much as possible, costly preprocessing steps that decrease the system's interactivity, such as those involved in the conversion or segmentation of a model.

In addition, the haptic controller should be robust and degrade gracefully as the limits of its performance are reached. Haptic devices require high controller servo rates—typically over 1000Hz—in order to achieve stability and high disturbance rejection. Failure to achieve these rates can lead to a system that is unstable, potentially causing device damage or user injury.

Finally, the haptic system should provide a high-level interface that hides many of the details of the haptic rendering process. Since graphic and haptic environments are often identical, it would be advantageous if the graphic and haptic specifications were similar.

This paper describes "HL," a new haptic interface library. The Application Programming Interface (API) of this library is almost identical to that of GL, the graphics hardware interface library of Silicon Graphics workstations. This allows haptic environments to be quickly and efficiently incorporated into graphics applications.

The HL library uses a multi-level control system to effectively simulate contacts with virtual environments. A key element of this control system is the notion of the virtual "proxy," similar to the "god-object" proposed by Zilles and Salisbury [27]. The virtual proxy is a representative object that substitutes for the physical finger or probe in the virtual environment. Figure 1 illustrates the motion of the virtual proxy, as the probe's position is altered. The motion of the proxy is akin to that of a robot greedily attempting to move toward a goal. When unobstructed, the robot moves directly towards the goal. When the robot encounters an obstacle, direct motion is not possible, but the robot 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

[1] Robotics Laboratory, Computer Science Department, Stanford University, Stanford, CA 94305
ruspini@cs.stanford.edu, khatib@cs.stanford.edu

[2] Interval Research Corporation 1801 Page Mill Road, Building C, Palo Alto, CA 94304
kolarov@interval.com

goal. When the robot is unable to decrease its distance to the goal, it stops at the local minimum configuration.

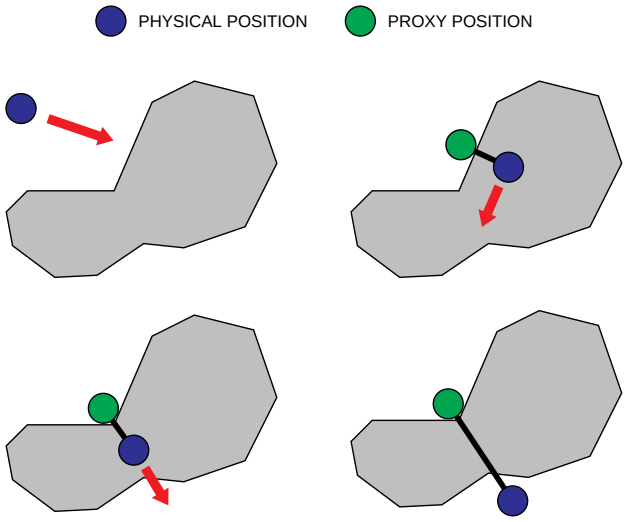

Figure 1: Virtual Proxy Example

Since a strong correspondence exists between the movement of the proxy and robot motion planning, many of the algorithms used in our implementation were developed originally for robotics applications. With this interaction model, the task of the haptic servo controller reduces to minimizing the error between the configuration of the proxy and the position of the haptic device. In effect, the haptic device is used to attempt to physically move the goal to the location of the proxy.

The remainder of this paper is organized as follows: In Section 2, we discuss previous work in haptic rendering. The basic algorithm employed to update the virtual proxy's position is presented in Section 3. In Section 4, we discuss the implementation of force shading—the haptic equivalent of Phong shading [21] in graphics—within the virtual proxy framework. Section 5 discusses methods to simulate static and dynamic friction, and other surface and atmospheric effects. An overview of the current implemented system is presented in Section 6, and the low-level haptic controller is presented in Section 7. Sections 8 and 9 are devoted to the presentation of results and the discussion of future work.

2 BACKGROUND AND RELATED WORK

In *penalty* methods, forces proportional to the amount of penetration into a virtual volume are applied to the haptic device. For simple geometries, like spheres and planes, the direction and amount of penetration are easy to determine. The simplicity of this approach has facilitated the study of many interesting situations such as those involving dynamic objects and surface effects. Massie and Salisbury extended this technique by subdividing the internal volume and associating each sub-volume with a surface toward which repulsion forces are exerted [17]. This approach has also been used successfully to allow haptic interactions with volumetric data [1,12].

These approaches, however, have a number of drawbacks. When multiple primitives touch or are allowed to intersect it is often difficult to determine which exterior surface should be associated with a given internal volume. In the worst case, a global search of all the primitives may be required to find the nearest exterior surface, as seen in Figure 2(a). In addition, as a finger probe penetrates a surface it will eventually become closer to another surface of the object. The resultant force actively pushes the probe out through this second surface. This situation is illustrated in Figure 2(b). Finally, as shown in Figure 2(c), small or thin objects may have insufficient internal volume to generate the constraint forces required to prevent the probe from passing through the obstacle. This problem is particularly troublesome in graphics applications since most graphic models are constructed almost exclusively from infinitely thin polygons, lines and points.

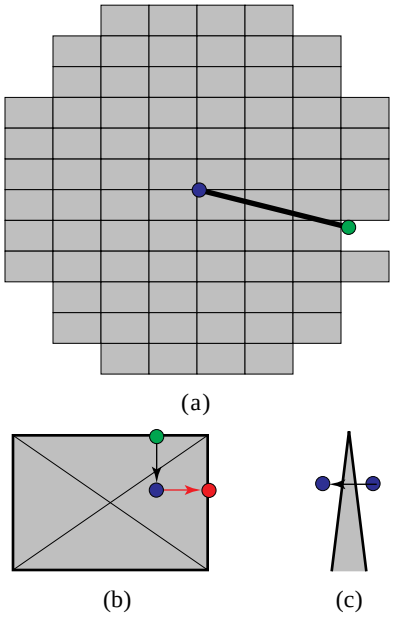

Figure 2: Limitations of "Penalty" based Haptic Rendering Methods. (a) Lack of locality: removal of primitive will create new nearest surface, (b) Force Discontinuities: application of force causes probe to be attracted toward other surfaces. (c) "Pop-Thru" of thin objects.

Constraint-based methods were first proposed for haptic applications by Zilles and Salisbury [27] to address the limitations of penalty-based approaches. These methods employ a god-object which, similar to the virtual proxy, is constrained by the objects in the environment. This approach has been used to model interactions between a point-size god-object and complex polygonal models. The virtual proxy is an extension of this idea. This paper presents how, in addition to surface constraints—force shading [19], friction, surface stiffness, and texture can be modeled by simply changing the position of the virtual proxy. Also, because the virtual proxy has a finite size, it does not slip through the tiny numerical gaps found in most polygonal meshes and can therefore operate without first having to reconstruct the topology of a surface as is required in the original god-object approach [27].

3 UPDATING PROXY POSITION

For simplicity, we will 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 should therefore be large enough to avoid

falling through the holes in the underlying model. In addition, the user will often wish to make the proxy large enough so that it is easily visible on a graphical display. We also assume that all the obstacles in the environment can be divided into a finite set of convex components.

During the update process, a goal configuration for the proxy is found at each time step and the proxy attempts to move to this configuration by direct linear motion. Initially, the goal configuration is the location of 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 and goal configurations, falls within one radius of any object in the environment. Since many primitive objects may exist in the environment, an efficient means of determining which primitives intersect the proxy's path is required. Several fast, general purpose, algorithms have been developed for this purpose [15,10]. In our current implementation, we employ an algorithm originally developed for path-planning applications [22] that builds a bounding-sphere hierarchy for each object and is capable of quickly finding the shortest distance between non-convex bodies.

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) [14], consist of all points within one proxy radius of the original obstacles. Note that, 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 then be found where the line segment that represents the proxy's path intersects the C-obstacle. An example of configuration space, C-obstacles, and proxy constraint planes is shown in Figure 3.

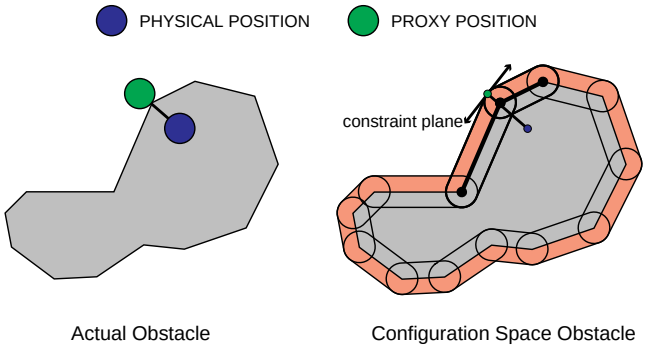

Figure 3: Configuration Space Obstacles & Constraint Planes

Introduction of the configuration space allows us to model the proxy as a point and the obstacles as uniquely defined local planar surfaces. The proxy is moved until it makes contact with the closest constraint plane. Planes that fall below this new position cannot affect the local motion of the proxy and may therefore be pruned. If the proxy reaches the user's position, no further movement is required. Otherwise, a new sub-goal is generated observing that 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 desired solution is the point within this convex region (the local free-space) that minimizes the distance to the user's position. Since this problem is independent of coordinate translation, and since all the constraint planes go through the current proxy position, the problem can be written compactly as

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

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

This problem may be solved using a standard quadratic programming package such as that introduced by Gill et. al [9]. In our case, however, there are many simplifications that make possible a simpler and faster solution. In our implementation, this problem is solved in two steps. The minimum set of active constraint planes is found first; this set is then used to find a new sub-goal position. If the desired solution lies on a face of the convex free space, then the solution lies on only one of the constraint planes. If the solution lies on an edge, then two constraint planes are required. If the solution lies on a vertex three planes are needed. Finally, if the user's position lies in the free space, then no constraint planes are required. This convex free space region has a dual space consisting of 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 on the convex hull of this region to a point $\hat{p}$, a unit vector with the same direction as p. This problem may be treated using the same algorithms employed in the collision detection process [8]. The vertices of the closest face, edge or vertex indicate that the corresponding constraint planes bound the solution. An example of this mapping is shown in Figure 4. As illustrated, the solution x is constrained by planes $\mathbf{a}, \mathbf{b}$ and the plane at infinity $\mathbf{o}$. In the dual, this corresponds to $\hat{p}$ being nearest the face $\{\mathbf{a}, \mathbf{b}, \mathbf{o}\}$.

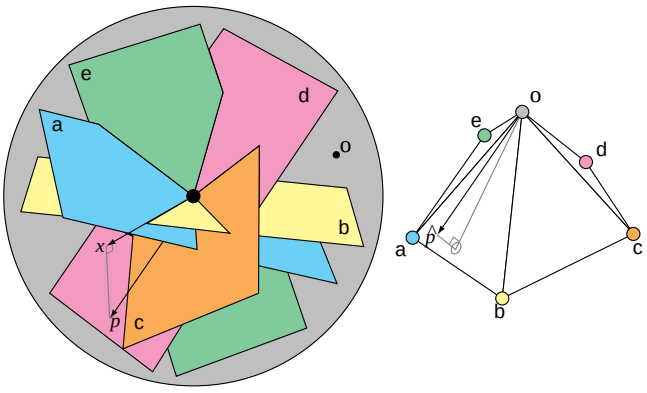

Figure 4: Constraint Planes and Equivalent Free Space Dual

Once the bounding planes have been determined, Equation 1 may be solved using only the active planes as constraints.

With the inequalities replaced by equalities, the problem can be solved easily using Lagrange multipliers as is described by Zilles in [27] to find 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. Once the new sub-goal is found, the iteration may continue.

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

4 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 [11,21]. Morgenbesser and Srinivasan [19] were the first to demonstrate that a similar haptic effect may be created. Their solution changes the direction of the normal force while retaining the magnitude caused by the penetration of the original polygonal surface. While this technique produces compelling shading effects, it is unclear how to extend this approach to deal with multiple intersecting shaded surfaces or additional surface effects, such as friction or texture.

Our force shading approach handles situations involving multiple intersecting, shaded surfaces that are in contact with the proxy at the same time. These situations arise, for example, when two force-shaded cylinders are placed side by side. In addition, the shading effect is created solely by moving the position of the proxy, thus helping to guarantee solution stability.

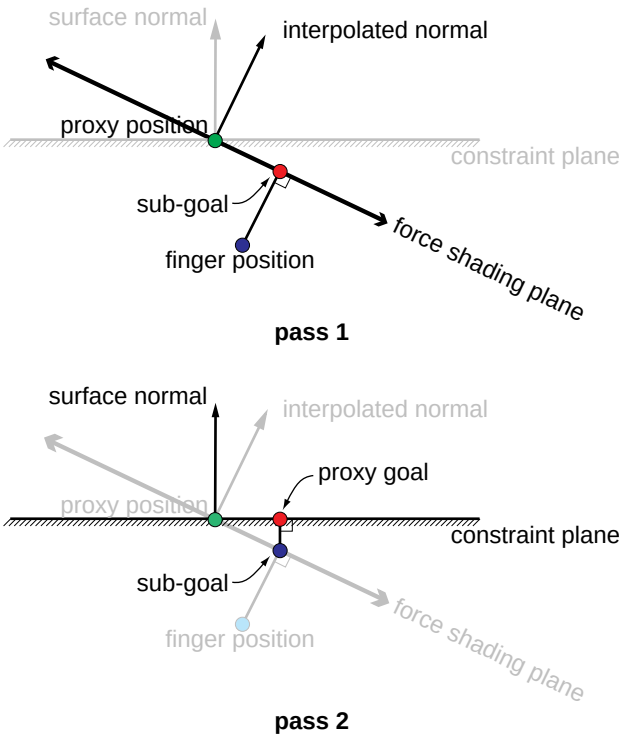

Figure 5: Two Pass Force Shading with Supplied Normals

When a polygonal surface with user specified normals is encountered a new local surface normal is calculated by interpolating the normals from the vertices of the polygon. Since the required interpolation weights have already been computed as part of the collision detection process, this determination requires little additional computation. This interpolation is very similar to that required for Phong shading in computer graphics applications [21]. 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 actual (non-interpolated) constraint planes. While this approach is slightly more computationally expensive than previous efforts [19,23], it properly 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 5.

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

Note that 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. In graphic models the interpolated and the true surface normals typically differ by less then 30°. In this case, the added energy is very small, and is not noticed by the user. In all of our tests the motion was stable.

The difference between the force shaded surface and a flat surface is illustrated in Figure 6. 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.

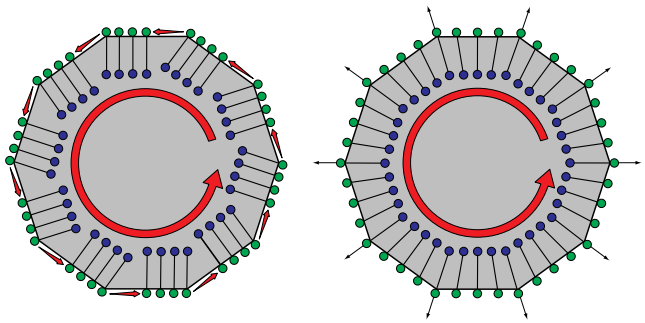

Figure 6: Effect of Flat (a) vs. Force Shaded (b) surface on proxy motion

As seen in Figure 6(a), a strong discontinuity occurs when the proxy finally reaches an obstacle edge. In Figure 6(b), surface normals have been specified on the vertices. The resulting movement of the proxy shows that the resultant force is always perpendicular to the interpolated circular object. This is what would be expected if the user were moving around a perfectly circular object. Although the proxy remains on the polygonal surface the motion of the proxy is continuous and therefore the resultant force feels smooth to the user.

5 SURFACE PROPERTIES

Several researchers [2,5,16,24,25] have proposed methods to simulate static, dynamic, viscous friction and texture. Unlike previous methods, our implementation creates all these effects solely by restricting the movement of the proxy. In this way the stability of the final solution can be better controlled.

5.1 Static Friction

Static friction (stiction) 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 $\|f_t\| \le \mu_s \|f_n\|$, i.e., the user's position is in the friction cone of the surface. When any constraint surface is in static contact with the proxy, the proxy's position is prevented from changing by making the new sub-goal position equal to the current proxy position.

5.2 Viscous and Dynamic Friction

Our approach for modeling viscous damping and Coulomb friction is based on the observation of 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

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

where b is a viscous damping term, and 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

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

This limit can be used to bound the amount that the proxy can travel in one clock cycle. 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. Note that this method does not require computation of the finger velocity and is therefore not susceptible to errors caused from trying to estimate this value from a finite number of encoder values.

In the majority of current treatments, the stiffness of a surface is modeled by reducing the position gain of the haptic controller. This approach is undesirable in our system since the location of the proxy models many complex and intermixed phenomena. In addition, it is desirable to keep the haptic controller at settings that are chosen to optimize its performance based solely on the inertia, friction and stiffness of the mechanical system and not on the needs of the virtual environment.

Given a surface with stiffness s, $0 \le s \le 1$, it is possible to change the apparent stiffness of a surface without altering any of the controller's parameters by choosing a new point p' such that

$$p' = v + s(p - v), \qquad (4)$$

where p is the position of the proxy assuming an infinitely stiff surface and v is the position of the user's finger. The point p' is used as the proxy position for the haptic control loop. The old proxy position is still retained to allow the proxy to continue to follow the surface of the object. This approach is based on the intuitive physical observation that, as pressure is applied, the surface of a soft body will indent, resulting in the finger penetrating the volume of the un-deformed object, as seen in Figure 7. Caution should be taken when adjacent surfaces have different stiffnesses. As the proxy moves from a non-rigid surface to one with a higher stiffness, the distance between the proxy and the finger will increase, adding energy to the system. The result is that the user feels what appears to be an active unnatural surface. A more realistic effect can be created by altering the polygonal surface as it is affected by forces applied by the user. The surface can be deformed to reveal properties of the model's internal structure that are not possible to create solely by altering the stiffness of the boundary surface.

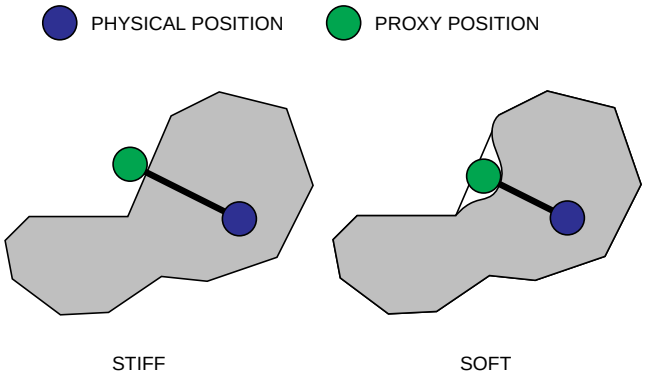

Figure 7: Physical Intuition Behind Stiffness Model

5.3 Texture

Haptic textures were first demonstrated by Minsky et al. [18] for the haptic display of height fields on a two degree of freedom planar haptic display. In our system, an 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 each clock cycle. This produces a convincing effect with slowly changing textures. Ideally, the texture values should be evaluated as the proxy moves along the surface of the object to ensure that a significant event is not missed as the proxy travels along that surface. This evaluation is required, for example, when layering a texture of thin high friction grid lines on a planar surface. Without evaluating the texture values along the path these grid-lines may be missed.

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 [4]. Our current texture bump-map technique can generate one additional constraint plane for each textured surface. This approach is adequate to model

continuously differentiable textured surfaces. Grooved or cratered surfaces which contain sharp edges and corners may require additional constraint planes and the monitoring of the motion of the proxy to ensure that it is not constrained by other surface features as it moves along the surface.

6 SYSTEM IMPLEMENTATION

Our current system runs on two computers: 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 important since the haptic servo loop must run at a very high rate, typically greater than 1000Hz, to achieve a high fidelity force display. Most application programs typically run at a much slower rate (~30Hz).

In our system 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. This is desirable since CPU processor performance is increasing rapidly while the latency of I/O connections has been largely stagnant. In our current system, a SGI workstation is used as the haptic client, a Pentium Pro PC is used as the server, and communication between them is performed over a regular ethernet connection via TCP/IP packets.

6.1 The Client Application

Applications communicate to the haptic server through the HL network interface library. 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. Objects are retained until over-written. Transformations are provided to allow objects and primitives to be freely translated or rotated. Surface normals and texture coordinates can be associated with polygonal vertices to allow for the specification of smooth 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 sequences of line segments are represented as a set 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 with 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 the object. The bounding representation, based on spheres, is similar to that first proposed by Quinlan [22].

This hierarchy of bounding spheres is constructed by first covering each polygon with small spheres in a manner similar to scan conversion in computer graphics. These spheres are the leaves of an approximately balanced binary tree. Each node of this tree represents a single sphere that completely contains all the leaves of its descendants. After covering the object, a divide and conquer strategy is used to build the interior nodes of the tree. This algorithm works in a manner similar to quick-sort. First an axis aligned bounding box that contains all the leaf spheres is found. The leaf spheres are then divided along the plane through the mid-point of the longest axes of the bounding box. Each of the resulting two subsets should be compact and contain approximately an equal number of leaf spheres. The bounding tree is constructed by recursively invoking the algorithm on each subset and then creating a new node with the two sub-trees as children. A cut-away view showing the leaf nodes (yellow) and bounding sphere hierarchy for a typical model is illustrated in Figure 8. Note that a node is not required to fully contain all the descendant internal nodes, only the descendant leaf nodes.

Figure 8: Cut-Away of the Bounding Hierarchy of a Cat Model

Two heuristics are used to compute the bounding sphere of a given node. The first heuristic finds the smallest bounding sphere that contains the spheres of its two children. The second method directly examines the leaf spheres. The center is taken as the mid-point of the bounding box already computed earlier. The radius is taken to be just large enough to contain all the descendant leaf nodes. The method that generates the sphere with the smallest radius is used for the given node. The first heuristic tends to work better near the leaves of the tree, while the second method produces better results closer to the root. This algorithm has an expected $O(n \lg n)$ execution time, where n is the number of leaf spheres.

7 HAPTIC CONTROLLER

Reliance on a virtual proxy reduces the task of the haptic servo controller to minimization of the error between the configuration of the proxy and position of the haptic device. Reducing position error of a mechanical system is a problem which has been discussed extensively in the robotics literature [6]. In our current implementation we rely on a simple operational space proportional derivative (PD) controller [13]. As all modeling effects are achieved by the movement of the proxy, controller gains and other parameters can be set by sole consideration of the properties of the mechanical system.

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 is easier to ensure and the fidelity of the haptic display degrades 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.

8 RESULTS

Our haptic library has been successfully tested on a large number of polygonal models, including some containing more than 24,000 polygonal primitives. In our tests the client computer was a SGI Indigo2 High Impact running IRIX 6.2 and 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 PHANToM manipulator. This 3-degree-of-freedom force-feedback device has sufficiently high stiffness, low inertia and low friction for high fidelity force display. The server produced stable results with position gains over 1800 Newtons/meter with no artificial damping. The proxy update loop computation time is approximately $O(\lg n)$ where n is the number of polygons. This slow asymptotic growth is the consequence of the dependence of the proxy's movement on only its local environment. In contrast, the rendering time for a graphic display, where the entire world may be visible at one time, is inherently $O(n)$.

The current system is adept in modeling a large number of geometric models. Some examples are shown in Figures 9,10 and 11. Figure 9 shows a VRML model of an AT-AT from Star Wars containing over 11,000 polygons. The high level interface simplifies the implementation of applications like VRML browsers. Figure 10 shows a VRML model of the classic teapot, composed of 3416 triangular surfaces. Force shading is used to model the apparently curved surfaces of the underlying polygonal model. Figure 11 shows a sample test application where the user can click virtual buttons to select a variety of geometrical models with numerous different surface characteristics. These models can be moved to make them contact or overlap one another, creating possibly thousands of unexpected new intersections, edges, and corners.

In all cases, the location of the virtual proxy, rather than finger position, is displayed to the user, further adding to the sense of rigidity of the modeled environment [26].

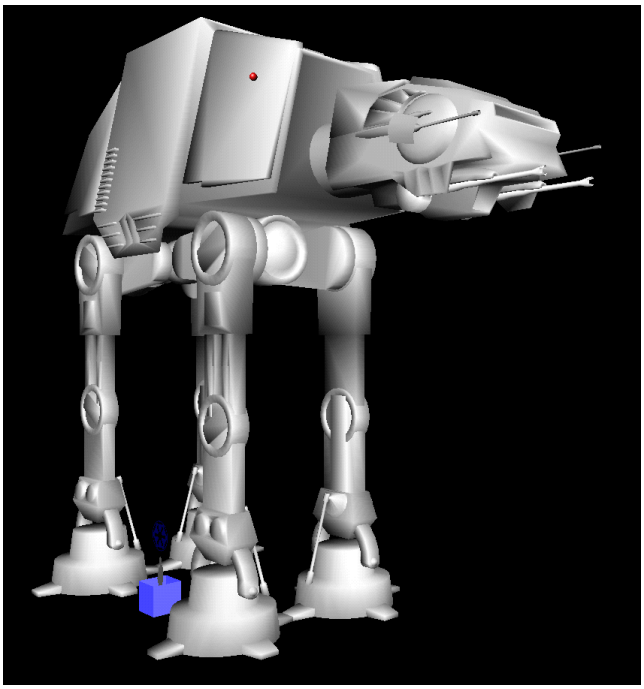

Figure 9: Haptic AT-AT (11088 polygons)

Figure 10: Force Shaded Teapot (3416 polygons)

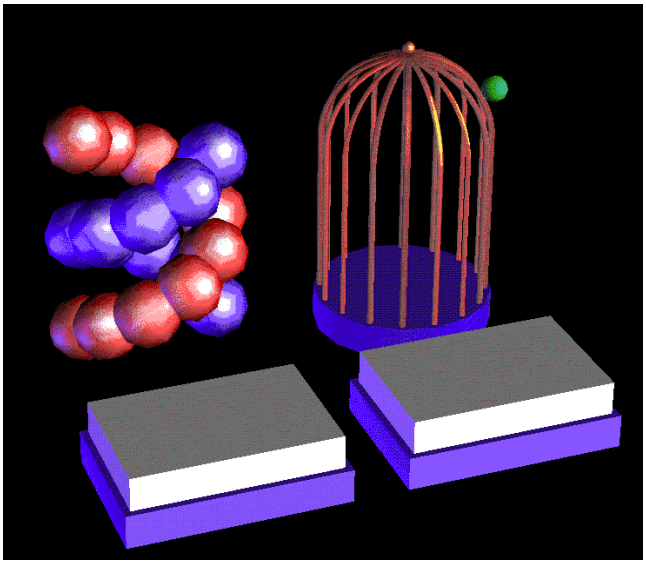

Figure 11: Interactive Haptic Environment

9 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 must be specified by the client processor, result in a discontinuous jerky motion. Furthermore, in some cases, it is possible 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.

Finally, the virtual proxy framework can be expanded to handle implicit surfaces like splines, or even volumetric information directly without first transforming these representations into a polygonal surface model. This ability is beneficial because often the cost of this transformation can be prohibitively expensive, or can greatly reduce the interactivity of the application.

CONCLUSION

The system presented in this paper is capable of the haptic simulation of complex graphical environments. The common virtual proxy framework, used for modeling all haptic effects, reduces the low-level control of the haptic device to a simple positional controller. This controller may operate in real time if its operation is separated from the rest of the haptic rendering process, thus helping to ensure that the haptic controller's stability. The high level user interface allows graphic applications to quickly and easily incorporate haptic technology and increases the ability to manipulate and interact with virtual environments.

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. is from Avalon Viewpoint archive. The reseach was supported by research grants from NASA/JSC, grant NGT-9-6, Boeing, Interval Research Corporation, and NSF, grant IRI-9320017.

REFERENCES

[1] Avila, R. S., Sobierajski, "A Haptic Interaction Method for Volume Visualization," *Visualization '96 Proceedings*, October 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] Baraff, D., "Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies," *SIGGRAPH 89 Proceedings*, (August 1989), pp. 223-232.

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

[5] 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.

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

[7] 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.

[8] 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.

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

[10] Gottschalk, S., Lin, M. C., Manocha D., "OBBTree: A Hierarchical Structure for Rapid Interference Detection," *SIGGRAPH 96 Proceedings*, (August 1996), pp. 171-180.

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

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

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

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

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

[16] 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 Proceedings*, (August 1996), pp. 447-452.

[17] 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, Nov. 6-11), vol. 1, pp.295-301.

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

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

[20] Ouh-Young, M., "Force Display in Molecular Docking," *Ph. D. Dissertation, University of North Carolina at Chapel Hill, UNC-CH CS TR90-004*, February, 1990.

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

[22] Quinlan, S., "Efficient Distance Computation between Non-Convex Objects," *Int. Conference on Robotics and Automation*, (April 1994).

[23] Ruspini, D., Kolarov, K., "Robust Haptic Display of Graphical Environments," Proc. of The First Phantom User's Group Workshop, September 1996

[24] 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.

[25] 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, pp. 123-130, April 1995.

[26] 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.

[27] 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.