6

# Haptic Rendering in Virtual Environments

Cagatay Basdogan[1] and Mandayam A. Srinivasan[2]
[1]*Jet Propulsion Laboratory*
*California Institute of Technology*
*Pasadena, CA 91109*
*Cagatay.Basdogan@jpl.nasa.gov*
[2]*Laboratory for Human and Machine Haptics (The Touch Lab)*
*Massachusetts Institute of Technology*
*Cambridge, MA 02139 USA*
*srini@mit.edu*

## 1. INTRODUCTION

The goal of *haptic rendering* is to enable a user to touch, feel, and manipulate virtual objects through a haptic interface. With the introduction of high-fidelity haptic devices (see chap. 5, this volume), it is now possible to simulate the feel of even fine surface textures on rigid complex shapes under dynamic conditions. Starting from the early 1990s, significant progress has occurred in our ability to model and simulate haptic interactions with three-dimensional (3-D) virtual objects in real time (Salisbury & Srinivasan, 1997; Srinivasan & Basdogan, 1997). The rapid increase in the number of workshops, conference sessions, community web pages, and electronic journals on haptic displays and rendering techniques* indicates growing interest in this exciting new area of research called *computer haptics*. Just as computer graphics is concerned with synthesizing and rendering visual images, computer haptics is the art and science of developing software algorithms that synthesize computer generated forces to be displayed to the user for perception and manipulation of virtual objects through touch. Various applications of computer haptics have been developed in the areas of medicine (e.g., surgical simulation, telemedicine, haptic user interfaces for blind people, rehabilitation of patients with neurological disorders; see chaps. 47–51, this volume), entertainment

*See the proceedings of Phantom Users Group Workshops starting from 1996 (http://www.ai.mit.edu/conferences/), ASME Dynamics Systems and Control (DSC) Division starting from 1993, IEEE International Conference on Robotics and Automation, and IEEE Virtual Reality Conference. Also visit the haptic community pages at http://www.sensable.com/community/index.htm and http://haptic.mech.nwu.edu/, and the hapticse-journal at http://www.haptics-e.org/.

(e.g., 3-D painting, character animation, morphing and sculpting; see chap. 55, this volume), mechanical design (e.g., path planning and assembly sequencing; see chap. 54, this volume), and scientific visualization (e.g., geophysical data analysis, molecular manipulation; see chap. 53, this volume). More applications are anticipated as devices and rendering techniques improve and computational power increases. This chapter will primarily focus on fundamental concepts of haptic rendering, with some discussion of implementation details. Although, it is impossible to cite all relevant work within the constraints of this chapter, an attempt has been made to cover the major references. Given that current technology is mature for net force and torque feedback (as in tool usage in the real world) and not tactile feedback (as in actual distribution of force fields on the skin during contact with real objects), the discussion is restricted to techniques concerning the former. In general, the concepts discussed in the chapter include: (1) *haptic interaction paradigms*, which define the nature of the "haptic cursor" and its interaction with virtual objects; and (2) *object property display algorithms*, which enable rendering of surface and material properties of objects and their behavior through repeated use of the haptic interaction paradigm.

## 2. PRINCIPLES OF HAPTIC RENDERING: OBJECT SHAPE

Typically, a haptic rendering algorithm is made of two parts: (a) collision detection; and (b) collision response (see Fig. 6.1). As a user manipulates the probe of a haptic device, the new position and orientation of the haptic probe are acquired and collisions with virtual objects are detected (i.e., *collision detection*). If a collision is detected, interaction forces are computed using preprogrammed rules for *collision response* and conveyed to the user through the haptic device to provide him or her with the tactual representation of 3-D objects and their surface details. Hence, a haptic loop, which updates forces around 1 kHz (otherwise, virtual surfaces feel softer, or, at worst, instead of a surface it feels as if the haptic device is vibrating), includes at least the following function calls:

- get_position (vector and position); // position and/or orientation of the end effector
- calculate_force (vector and force); // user-defined function to calculate forces
- send_force (vector force); // calculate joint torques and reflect forces back to the user

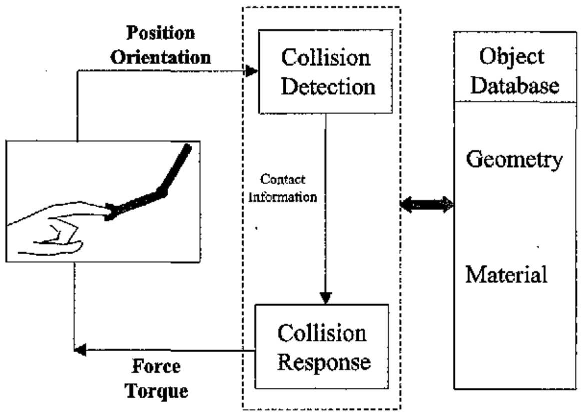
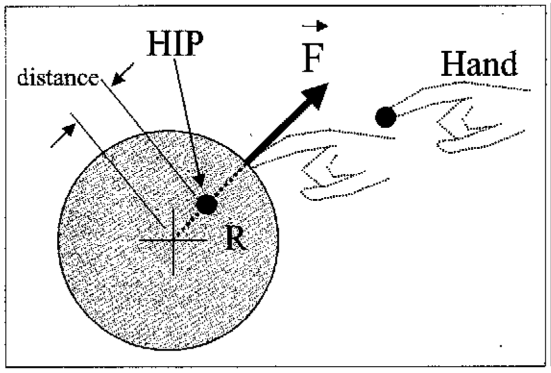


FIG. 6.1. A haptic interaction algorithm. Typically made of two parts: (a) collision detection; and (b) collision response. The haptic loop seen in the figure requires an update rate of around 1 kHz for stable force interactions. Computationally fast collision detection and response techniques are required to accommodate this requirement.



```
void calculate_force (Vector &force)
    {
        float X, Y, Z, distance;
        float R = 20.0;

        X = HIP[0]; Y = HIP[1];  Z = HIP[2];
        distance = sqrt(X*X + Y*Y + Z*Z);

        if(distance < R)  //collision check
        {
            force[0] = X/distance * (R-distance);
            force[1] = Y/distance * (R-distance);
            force[2] = Z/distance * (R-distance);

        }

    }
```

FIG. 6.2. Haptic rendering of a 3-D sphere in virtual environments. (The software code presented on the right-hand side calculates the direction and the magnitude of the reaction force for the sphere discussed in the example. The sphere has a radius of 20 units and is located at the origin of a 3-D virtual space.)

To describe the basic concepts of haptic rendering, consider a simple example, haptic rendering of a 3-D frictionless sphere, located at the origin of a 3-D virtual space (see Fig. 6.2). Assume that the user can only interact with the virtual sphere through a single point, which is the end point of the haptic probe, also known as the *haptic interaction point* (HIP). In the real world, this is analogous to feeling the sphere with the tip of a stick. As the 3-D space is freely explored with the haptic probe, the haptic device will be *passive* and will not reflect any force to the user until a contact occurs. Since the virtual sphere has a finite stiffness, HIP will penetrate into the sphere at the contact point. Once the penetration into the virtual sphere is detected and appropriate forces to be reflected back to the user are computed, the device will become *active* and reflect opposing forces to the user's hand to resist further penetration. The magnitude of the reaction force can easily be computed by assuming that it is proportional to the depth of penetration. Assuming no friction, the direction of this force will be along the surface normal as shown in Fig. 6.2.

As it can be seen from the example given above, a rigid virtual surface can be modeled as an elastic element. Then, the opposing force acting on the user during the interaction will be:

$$\vec{F} = k\Delta\vec{x}$$

where $k$ is the stiffness coefficient and $|\Delta\vec{x}|$ is the depth of penetration. Whereas keeping the stiffness coefficient low would make the surface perceived soft, setting a high value would make interactions unstable by causing undesirable vibrations. Figure 6.3 depicts changes in force profile with respect to position for real and virtual walls. Since the position of the probe tip is sampled digitally with a certain frequency during the simulation of a virtual wall, a "staircase" effect is observed. This staircase effect leads to energy generation (see the discussions and suggested solutions in Colgate & Brown, 1994, and Ellis, Sarkar, & Jenkins, 1996).

Although the basic recipe for haptic rendering of virtual objects seems easy to follow, rendering of complex 3-D surfaces and volumetric objects requires more sophisticated algorithms than the one presented for the sphere. The stringent requirement of updating forces around 1 kHz leaves very little CPU time for computing collisions and reflecting forces back to the user in real time when interacting with complex shaped objects. In addition, the algorithm given above for rendering of a sphere considered only "point-based" interactions (as if interacting with objects through the tip of a stick in the real world), which is far from what our hands are capable of in the real world. However, several haptic rendering techniques have been developed recently to simulate complex touch interactions in virtual environments (reviewed

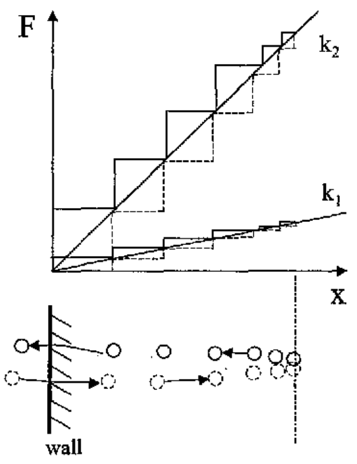


FIG. 6.3. Force-displacement curves for touch interactions with real and virtual walls. In the case of real walls, the force-displacement curve is continuous. However, a "staircase" effect is seen when simulating touch interactions with a virtual wall. This is due to the fact that haptic devices can only sample position information with a finite frequency. The difference in the areas enclosed by the curves that correspond to penetrating into and out of the virtual wall is a manifestation of energy gain. This energy gain leads to instabilities as the stiffness coefficient is increased (compare the energy gains for stiffness coefficients $k_2$ and $k_1$). On the other hand, a low value of the stiffness coefficient generates a soft virtual wall, which is not desirable either.

in Hollerbach & Johnson, 2001; Salisbury & Srinivasan, 1997; Srinivasan & Basdogan, 1997). The existing techniques for haptic rendering with force display can be distinguished based on the way the probing object is modeled: (1) a point (Adachi, Kumano, & Ogino, 1995; Avila & Sobierajski, 1996; Ruspini, Kalarov, & Khatib, 1997; Ho, Basdogan, & Srinivasan, 1999; Zilles & Salisbury, 1995); (2) a line segment (Basdogan, et al., 1997; Ho et al., 2000); or (3) a 3-D object made of a group of points, line segments, and polygons (McNeely et al., 1999). The type of interaction method used in simulations depends on the application.

In point-based haptic interactions, only the end point of the haptic device (i.e., the HIP) interacts with virtual objects (Fig. 6.4b). Each time a user moves the generic probe of the haptic device, the collision detection algorithm checks to see if the end point is inside the virtual object. If so, the depth of indentation is calculated as the distance between the current HIP and the corresponding surface point, also known as the *ideal haptic interface point* (IHIP; also called god-object, proxy point, or surface contact point; see Fig. 6.5). For exploring the shape and surface properties of objects in VEs, point-based methods are probably sufficient and could provide users with similar force feedback as what they would feel when exploring objects in

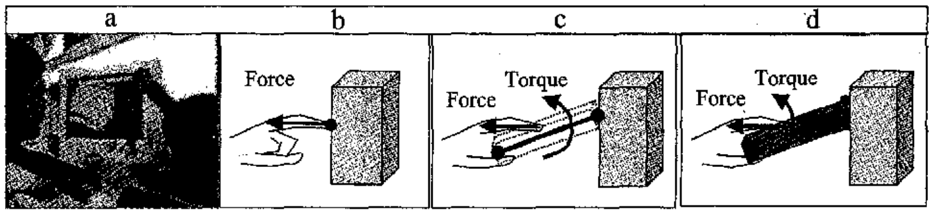


FIG. 6.4. Existing interaction methods for haptic rendering with force display can be distinguished based on the way the probing object is modeled: (b) a point; (c) a line segment; and (d) a 3-D object.

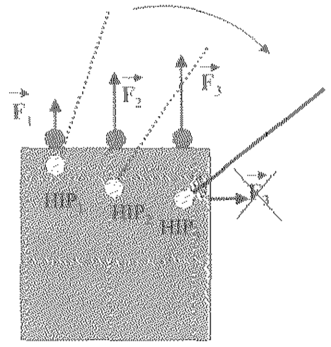


FIG. 6.5. A haptic rendering algorithm that is based on point interactions must calculate the IHIP for a given HIP at each rendering cycle. (IHIP can be considered as the surface point of the object with which the tool or finger would be in contact. The computation of this point relies on contact history. In the figure, for example, ignoring contact history and always choosing the closest point on the object surface as the new IHIP for a given HIP would make the user feel as if he is pushed out of the object as HIP moves from point 2 to point 3.

real environments with the tip of a stick. Using a point-based rendering technique, polyhedrons (Ho et al., 1999; Ruspini et al., 1997; Zilles & Salisbury, 1995). NURBS (Stewart, Chen, & Buttolo, 1997; Thompson, Johnson, & Cohen, 1997), implicit surfaces (Salisbury & Tarr, 1997), and volumetric objects (Avila & Sobierajski, 1996) have been successfully rendered. Point-based methods, however, are not capable of simulating more general *tool–object* interactions that involve single or multiple objects in contact with the tool at arbitrary locations of the tool. In such a context, both forces and torques displayed to the user need to be independently computed.

In ray-based interactions, the generic probe of the haptic device is modeled as a line segment whose orientation is taken into account, and collisions are checked between the finite line and objects. This approach enables the user to touch multiple objects simultaneously. In addition to forces, torque interactions can be simulated, which is not possible with point-based approaches (see Fig. 6.4c). Ray-based rendering can be considered as an approximation to long tools and as an intermediate stage in progress towards the full interaction between a 3-D cursor and 3-D objects. Also, if the geometric model of the probing object can be simplified to a set of connected line segments, then the ray-based rendering technique can be used and will be faster than simulation of full 3-D object interactions. For example, haptic interactions between a mechanical shaft and an engine block have been successfully simulated (Ho et al., 2000), as well as the interactions between laparoscopic surgical instruments and deformable objects (Basdogan, et al. 1998; Basdogan, Ho, & Srinivasan, 2001) using multiple line segments and the ray-based rendering technique. However, if the probing object has a complex geometry and cannot be easily modeled using a set of line segments, simulation of 6 degrees of freedom (DOF) object–object interactions has to be considered.

Simulation of haptic interactions between two 3-D polyhedra is desirable for many applications, but this is computationally more expensive than point-based and ray-based interactions (see Fig. 6.4d). Although a single point is not sufficient for simulating force and torque interactions between two 3-D virtual objects, a group of points, distributed over the surface of a probing object, has been shown to be a feasible solution. For example, McNeely et al. (1999) simulated touch interactions between the 3-D model of a teapot and a mechanical assembly.

Irrespective of the interaction method used, a haptic rendering algorithm must include both collision detection and collision response computations (see Fig. 6.1). Although collision detection has been extensively studied in computer graphics (Cohen, Lin, Manocha, & Ponamgi, 1995; Hubbard, 1995; Gottschalk, Lin, & Manocha, 1996; Lin, 1993), existing algorithms are not designed to work directly with haptic devices to simulate touch interactions in virtual environments. Moreover, merely detecting collisions between 3-D objects is not enough for simulating haptic interactions. How a collision occurs and how it evolves over time (i.e., *contact history*) are crucial factors (see the discussion in Fig. 6.5 on contact history) in haptic rendering to accurately compute the interaction forces that will be reflected to the user through a haptic device (Ho et al., 1999; Ho, Basdogan, & Srinivasan, 2000).

Although collision detection algorithms developed in computer graphics cannot be used in haptics directly, several concepts developed for fast collision detection can be ported to haptics. For example, haptic rendering algorithms can easily take advantage of: (1) space partitioning techniques (i.e., partitioning the space that encloses the object into smaller subspaces in advance for faster detection of first contact); (2) local search approaches (i.e., searching only the neighboring primitives for possible new contacts); and (3) hierarchical data structures (i.e., constructing hierarchical links between the primitives that make up the object for faster access to the contacted primitive). In addition to these improvements, a client–server model has to be considered to synchronize visual and haptic displays for achieving faster update rates. Using multithreading techniques, for example, one can calculate the forces at 1 kHz in one thread while updating the visual images at 30 Hz in another thread (Ho et al., 1999). If the forces cannot be computed at 1 kHz, a numerical scheme can be developed to extrapolate the new forces based on the previously computed ones to maintain the constant update rate of 1 kHz for stable interaction (Basdogan, 2000; Ellis, Sarkar, & Jenkins, 1996).

## 3. RENDERING OF SURFACE DETAILS: SMOOTH SHAPES, FRICTION AND TEXTURE

Quite frequently, it is desirable to display object surfaces as smooth and continuous, even when an underlying polyhedral representation is employed. In computer graphics, for example, illumination models are used to compute the surface color at a given point of a 3-D object (Foley, van Dam, Feiner, & Hughes, 1995). Then, shading algorithms are applied to shade each polygon by interpolating the light intensities (Gouraud shading) or surface normals (Phong shading). Shading algorithms make the shared edges of the adjacent polygons invisible and provide the user with the display of visually smooth surfaces (Watt & Watt, 1992). One can integrate a similar approach into the study of haptics to convey the feel of haptically smooth object surfaces (Basdogan et al., 1997; Fukui, 1996; Morgenbesser & Srinivasan, 1996; Ruspini et al., 1997). By using the *force-shading* technique suggested by Morgenbesser and Srinivasan (1996), force discontinuities can be reduced and the edges of polyhedral object can be made to feel smoother. To implement force shading with polygonal surfaces (Basdogan et al., 1997), the surface normal at each vertex is precomputed by averaging the surface normals of neighboring polygons, weighted by their neighboring angles. During real-time computations, the collision point that divides the contacted triangle into three subtriangles is first detected. Then the surface normal ($\vec{N}_s$) can be calculated at the collision point by averaging the normals of the vertices ($\vec{N}_i$) of the contacted polygon, weighted by the areas $A_i$ of the three subtriangles:

$$\vec{N}_s = \frac{\sum_i^3 A_i \vec{N}_i}{\sum_i^3 A_i}$$

Haptic simulation of surface details such as friction and texture significantly improves the realism of virtual worlds. For example, friction is almost impossible to avoid in real-life

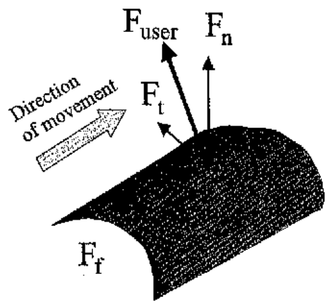


FIG. 6.6. Forces acting on the user ($F_{user} = F_n + F_t + F_f$) during the haptic simulation of friction and textures. (The normal force can be computed using a simple physics-based model such as Hooke's law ($F_n = k\,\Delta x$, where $|\Delta x|$ is the depth of penetration of the haptic probe into the virtual surface). To simulate coulomb friction, a force is created ($F_f = \mu F_n$, where $\mu$ is the coefficient of friction) that is opposite to the direction of the movement. To simulate textures, the magnitude and direction of the normal force vector are changed using the gradient vector of the texture field at the contact point.

and virtual surfaces without friction feel "icy-smooth" when they are explored with a haptic device. Similarly, most surfaces in nature are covered with some type of texture that is sensed and distinguished quite well by our tactile system (Srinivasan, Whitehouse, & LaMotte, 1990). Haptic texture is a combination of small-scale variations in surface geometry and its adhesive and frictional characteristics. However, displaying detailed geometry of textures would be computationally too expensive. Instead, both friction and texture can be simulated by appropriate perturbations of the reaction force vector computed using nominal object geometry and material properties. The major difference between friction and texture simulation via a haptic device is that the friction model creates only forces tangential to the nominal surface in a direction opposite to the probe motion, while the texture model can generate both tangential and normal forces in any direction (see Fig. 6.6).

Salcudean and Vlaar (1994) and Salisbury, Brock, Massie, Swarup, & Zilles, (1995) simulated static and dynamic friction such that the user feels the stick-slip phenomenon when the stylus of a haptic device is stroked over the surface of an object. By changing the mean value of the friction coefficient and its variation, more sophisticated frictional surfaces, such as periodic ones (Ho et al., 1999), and various grades of sandpaper (Green & Salisbury, 1997) can be simulated as well.

Haptic perception and display of textures in VEs require a thorough investigation, primarily because the textures in nature come in various forms. Luckily, graphics texturing has been studied extensively, and researchers can draw from that experience to simulate haptic textures in virtual environments. For example, bump mapping is a well-known technique in graphics for generating the appearance of a nonsmooth surface by perturbing surface normals. Blinn (1978) initially proposed this technique as a method of graphically rendering 3-D bumps for modeling clouds. His formulation depends on parameterization of the surface. Although the surface parameterization techniques have some advantages, they may generate uneven bump textures on the surface of complex shapes. Max and Becker (1994) suggested a direct method of mapping that does not require a transformation from global to parametric space. They developed a formulation that is purely based on the original surface normal and the local gradient of the height field that generates bumps on the surface of an object. Max and Becker utilized this technique to generate bumps on cloud contour surfaces. We used the same approach and perturbed the direction and magnitude of the surface normal ($\vec{N}_s$) to generate bumpy or textured surfaces that can be sensed tactually by users in virtual environments (Basdogan et al., 1997). The perturbed surface normals ($\vec{M}$) can be computed using the following formulation:

$$\vec{M} = \vec{N}_s - \nabla h + (\nabla h \cdot \vec{N}_s)\vec{N}_s$$
$$\nabla h = \frac{\partial h}{\partial x}\hat{i} + \frac{\partial h}{\partial y}\hat{j} + \frac{\partial h}{\partial z}\hat{k}$$

where, h(x,y,z) represents the texture field function and $\nabla h$ is the local gradient vector. One of the earliest methods of texture rendering that uses the force perturbation concept with a 2-DOF haptic device was developed by Minsky, Ming, Steele, Brook, & Behensky (1990). The formulation given above extends this concept to 3-D surfaces. In general, haptic texturing techniques can be classified into two parts: image-based and procedural.

## 3.1 Image-based Haptic Texturing

This class of haptic texturing deals with constructing a texture field from two-dimensional (2-D) image data. In computer graphics, digital images are wrapped around 3-D objects to make them look more realistic. While a graphical texture consists of 2-D texels with only color or gray scale intensities, a haptic texture should consist of texels with a height value. To display image-based haptic textures, the two-stage texture mapping technique of computer graphics (Bier & Sloan, 1986; Watt & Watt, 1992) can be followed. The first stage in this technique is to map the 2-D image to a simple intermediate surface such as a plane, cube, cylinder, or sphere. The second-stage maps the texels from the intermediate surface to the object surface. Following this stage, one can easily access the height value at any point on the object surface. In addition, the gradient vector at any point can be estimated using a finite difference technique and interpolation scheme. Then, the force perturbation concept described above can be used to display image-based haptic textures (see Ho et al., 1999, for implementation details).

## 3.2 Procedural Haptic Texturing

The goal of procedural haptic texturing is to generate synthetic textures using a mathematical function. The function usually takes the coordinate (x,y,z) as the input and returns the height value and its gradient as the output. For example, several investigators have implemented the well-known noise texture (Ebert et al., 1994; Perlin, 1985) to generate stochastic haptic textures (Basdogan et al., 1997; Fritz & Barner, 1996; Siira & Pai, 1996). Fractals are also appropriate for modeling natural textures since many objects seem to exhibit self-similarity (Mandelbrot, 1982). The fractal concept has been used in combination with other texturing functions, such as Fourier series and pink noise in various frequency and amplitude scales, to generate more sophisticated surface details (Ho et al., 1999). Similarly, several types of other graphical textures can be converted to haptic textures. For example, Ho et al. (1999) implemented haptic versions of reaction–diffusion textures (Turk, 1991; Witkin & Kass, 1991), spot noise (Wijk, 1991), and cellular texture (Worley, 1996).

## 4. RENDERING OF DEFORMABLE OBJECTS

Graphical display of deformable objects has been extensively studied in computer graphics. With the addition of haptic feedback, deformable objects have gained a new characteristic. Now, deformable models need to estimate not only the direction and amount of deformation of each node of an object but also the magnitude and direction of interaction forces that will be reflected to the user via a haptic device.

One way to categorize deformation techniques is according to the approach followed by researchers to deform surfaces: *geometry-* or *physics-based* deformations. In geometry-based deformations, the object or surrounding space is deformed, based purely on geometric manipulations. In general, the user manipulates vertices or control points that surround the 3-D object to modify the shape of the object. In contrast, physics-based deformation techniques aim to model the physics involved in the motion and dynamics of interactions. These models simulate physical behavior of objects under the effect of external and internal forces. Geometry-based deformation techniques are faster and are relatively easier to implement, but they do not necessarily simulate the underlying mechanics of deformation. Hence, the emphasis is on visual display and the goal is to make deformations more easily controllable and appear smooth to the user. Physics-based approaches, although necessary for simulating realistic behavior of deformable objects, are computationally expensive and not always suitable for real-time applications due to current limitations in computational power. In general, hybrid approaches that take advantage of both worlds seem to work well for many real-time applications.

Geometry- and physics-based deformation techniques used to render force-reflecting deformable objects can be further subgrouped as follows (see Basdogan, 1999, 2000, for more details):

1. Geometry-based Deformation Models

   • *Vertex-based:* The vertices of the object are manipulated to display the visual deformations. The reaction force can be modeled using Hooke's law, where depth of penetration can be computed based on the current and home positions of the vertex that is closest to the contact point. For example, in displaying soft tissue deformations graphically, polynomial functions that fit experimental data on finger pad deformation (Srinivasan, 1989) were used to remap the vertices of other organs (Basdogan, Ho, Srinivasan, Small, & Dawson, 1998).

   • *Spline-based:* Instead of directly transforming the vertices of the object, control points are assigned to a group of vertices and are manipulated to achieve smoother deformations. The concept of free-form deformation was originally suggested by Sederberg and Parry (1986) and extended by Hsu, Hughes, & Kaufman (1992) to direct free-form manipulation. The extension of this technique to haptic display of deformable objects with applications in medical simulation (Basdogan et al., 1998), computer-aided design (CAD), and haptic sculpting (Dachille, Qin, Kaufman, & El-sana, 1999; Edwards & Luecke, 1996) can be found in the literature. Here, the key advantages of using force feedback are to increase intuition, control deformations, and support implementation of various constraints.

2. *Physics-based* Deformation Models

   In comparison to geometry-based models, interaction forces are always part of the computation in physics-based models, and the developer does not need to consider a separate model for computing forces.

   • *Particle-based:* Particle systems consist of a set of point masses, connected to each other through a network of springs and dampers, moving under the influence of internal and external forces (see Witkin, Barraff, & Kass, 1998, for implementation details). In this model, each particle is represented by its own mass, position, velocity, and acceleration. This technique has been used extensively by computer graphics researchers in simulation of soft tissue and cloth behavior (Cover et al., 1993; Lee, Terzopoulos, & Waters, 1995; Ng & Grimsdale, 1996). Swarup (1995) demonstrated the implementation of this technique to haptic simulation of deformable objects.

   • *Finite element–based:* The volume occupied by a 3-D object is divided into finite elements, properties of each element are formulated, and the elements are assembled together to study deformation states for the given loads (Bathe, 1996). Due to the limited computational power that is available today, modeling simplifications have to be made to implement real-time finite element method with haptic displays (see Cotin, Delingette, & Ayache, 1999; James & Pai, 1999, for static computations; see Basdogan et al., 2001, for dynamic computations). For example, De and Srinivasan (1999) have proposed a modeling approach in which organs are viewed as thin walled structures enclosing a fluid, thereby converting the 3-D problem into an effectively 2-D one (see also Balaniuk & Costa, 2000). In Basdogan et al. (2001), to compute dynamical deformations and interaction forces, a modal analysis approach was implemented such that only the most significant vibration modes of the object were selected. In this study, interactions between two deformable objects were also incorporated, one finite element based and the other particle based. Basdogan (2001) has also recently proposed the spectral Lanczos decomposition method to obtain explicit solutions of the finite element equations that govern the dynamics of deformations. Both methods (modal analysis and spectral Lanczos decomposition) rely on modeling approximations but generate dynamical solutions that are computationally much faster than the ones obtained through direct numerical integration techniques.

- *Meshless methods:* Because of the complexities of mesh generation and consequent constraints imposed on the computations, a meshless numerical technique called the method of finite spheres has recently been applied to physics-based soft-tissue simulation (De, Kim, & Srinivasan, 2001).

In earlier studies, a loose coupling between force and displacement computations has been proposed to take advantage of human perceptual limitations (Basdogan et al., 1998). In this approach, vertex-based or spline-based approaches have been used to generate smooth visual deformations, whereas interaction forces were computed and reflected to the user based on a simple spring and a damper model between the new and home positions of the contact point. Because human perception of position and motion is dominantly influenced by visual cues in interacting with objects (Srinivasan, Beauregard, & Brock, 1996), the loose coupling between force and displacement computations was not readily sensed by users during simulations (see Model A in Fig. 6.7). However, this architecture is limited to simulation of geometric-based deformations (e.g., sculpting and animation). Since force and displacement computations are

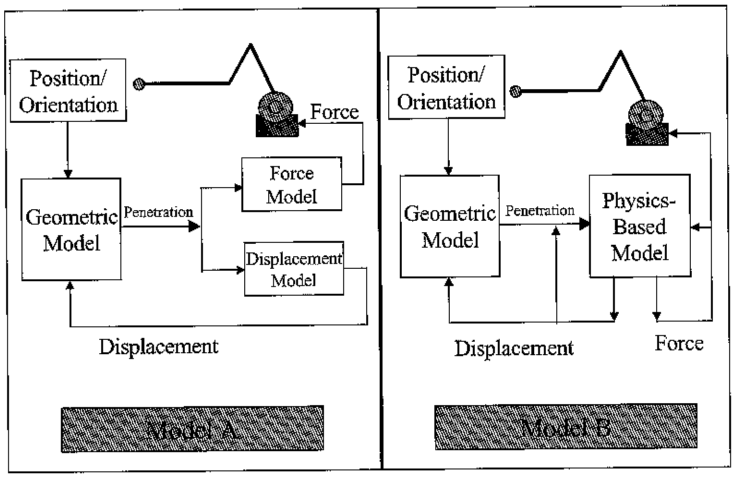


FIG. 6.7. Computational architectures for simulating force-reflecting deformable objects in virtual environments. In model A, force and displacement computations are loosely coupled. In model B, the physics-based model generates forces and displacements simultaneously, which are then fed back to the model to generate new values. Although this architecture is computationally expensive to implement, it is more comprehensive than architecture A (Basdogan, 1999).

usually tightly coupled in more physically based systems, a closed-loop architecture has to be developed to solve the equations that govern the physics of deformable behavior (see Model B in Fig. 6.7). In this architecture, forces and displacements computed in the previous cycle are continuously supplied back to the physics-based model to generate new values in the next cycle.

## 5. RENDERING OF DYNAMIC OBJECTS

Simulating haptic interactions with 3-D objects that translate and rotate over time is another challenge, which becomes computationally quite intensive if objects collide with each other, as well as with the haptic probe in real time. Dynamic equations have to be solved and the state of each object has to be updated at each iteration (see Fig. 6.8). Techniques to calculate forces and torques based on principles of rigid body dynamics (Baraff, 1994) and impulse mechanics (Mirtich & Canny, 1995; Mirtich, 1996) have been developed for graphical simulation of floating multiple objects. Latimer (1997) discusses the extension of these techniques to the haptics domain, whereas Colgate and Brown (1994) and Adams and Hannaford (1998) address the stability issues.

To describe the basic concepts of haptic interaction with dynamic objects, consider an example (see Fig. 6.8a). Assume that interactions are point-based again such that the user controls the dynamics of a floating object via only the tip point of the haptic probe. The new position and orientation of the object can be calculated using equations that govern the dynamics of rigid body motion. These equations can be solved in real time using a numerical

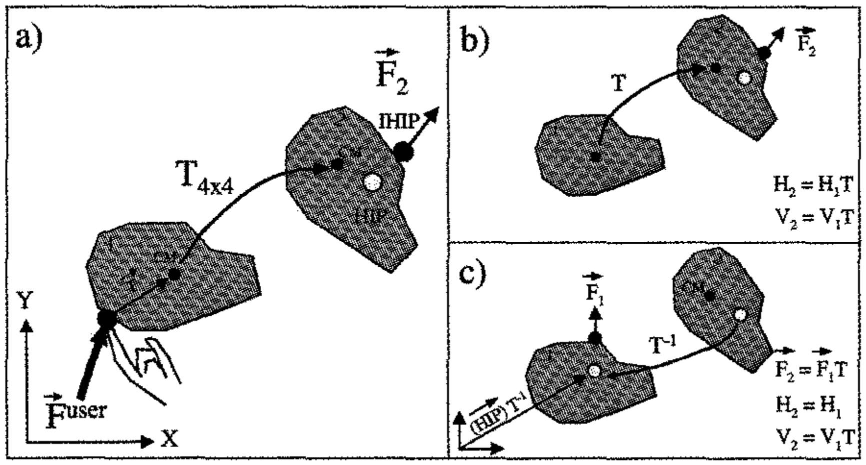


FIG. 6.8. Modeling point-based haptic interactions with a floating rigid object in virtual environments. (In the figure, T is the transformation matrix, $\vec{F}_2$ represents the force acting on the user through a haptic display, H and V represent the haptic and visual coordinates of the object respectively. Here, two different methods for computing the interaction forces when the object is transformed from State 1 to State 2 are discussed. Figure (b) describes a method in which both haptic and visual coordinates are updated and collisions are detected with the new haptic coordinates of the object to compute the new interaction force ($\vec{F}_2$) at State 2. In Figure (c), only visual coordinates are updated. First, HIP is multiplied with the inverse of the transformation matrix and collisions are detected with the original haptic coordinates. Then, the reaction force ($\vec{F}_1$) is computed relative to the original coordinates and then multiplied with the T matrix ($\vec{F}_2 = \vec{F}_1 \cdot \mathrm{T}$) to include the effects of transformation. Although both methods (b and c) are valid and return the same interaction force ($\vec{F}_2$) at the end, the method described in Figure (c) is computationally less expensive than the one described in Figure (b), because the haptic coordinates of the object are not required to be updated at each iteration when the object is transformed to a new state.

integration method, such as Euler integration, to update the state of the object at each cycle of the haptic loop:

| Constants: | Initial Conditions: | Integration over interval $\Delta t$: | Update Auxillary Quantities: |
|---|---|---|---|
| $I_{body}^{-1}, M$ | $\vec{v}_{t=0.0}$<br>$\vec{p}_{t=0.0}$<br>$R_{t=0.0}$<br>$L_{t=0.0}$<br>$\vec{\omega}_{t=0.0} = I_{body}^{-1}\vec{L}_{t=0.0}$ | $\vec{v}_{t+\Delta t} = \vec{v}_t + \Delta t(M^{-1}\vec{F}_t^{user})$<br>$\vec{p}_{t+\Delta t} = \vec{p}_t + \Delta t\,\vec{v}_{t+\Delta t}$<br>$R_{t+\Delta t} = R_t + \Delta t\omega^* R_t$<br>$\vec{L}_{t+\Delta t} = \vec{L}_t + \Delta t\vec{T}_t^{user}$<br><br>where:<br>$\omega^* = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$ | $I_{t+\Delta t}^{-1} = R_t I_{body}^{-1} R_t^T$<br><br>$\vec{\omega}_{t+\Delta t} = I_{t+\Delta t}^{-1}\,\vec{L}_{t+\Delta t}$ |

where, $\vec{F}^{user}$ and $\vec{T}^{user}$ represent the total force and torque acting on the object, $\vec{v}$ and $\vec{p}$ represent the linear velocity and position vectors and $\vec{\omega}$ and $\vec{L}$ are the angular velocity and momentum vectors. In addition, $R$, $M$, and $I$ are the rotation, mass and inertia matrices of the object respectively and $\Delta t$ is the sampling time (e.g. $\Delta t = 0.001$ seconds if the haptic loop is updated at 1 kHz). The total force acting on the object ($\vec{F}^{user}$) is calculated based on the depth of penetration of the haptic probe into the object. The torque acting on the object will be the cross product of $\vec{r}$ and $\vec{F}^{user}$ vectors (see Fig. 6.8). The details of the formulation and the computation of mass and inertia matrices can be found in Baraff (1997).

Since the position and orientation of the object change at each time step, the updated coordinates of the object should be used to detect collisions with the new coordinates of the HIP. However, it would be computationally too expensive to update the object database at each time step to detect collisions with the new HIP. A better strategy is to compute everything relative to the original coordinates and then apply the effects of transformation later (see the description in the legend for Fig. 6.8).

## 6. RECORDING AND PLAYING-BACK HAPTIC STIMULI

The concept of recording and playing back haptic stimuli (MacLean, 1996) to the user has some interesting applications. For example, the user equipped with a desktop haptic device may touch and feel the prerecorded textures of a car seat that is advertised by a car company over the Internet (this concept is similar to downloading a movie file and playing it on screen). In another application, preprogrammed haptic devices can be used in neuro-rehabilitation (see chaps. 49 and 50, this volume) to improve the motor-control skills of patients with neurological disorders by displaying prerecorded haptic trajectories (Krebs et al., 1998). In both of these examples, the user will be a passive participant of the system, though he or she may have some control over the device. As the user holds the end-effector of the device gently, the device guides his or her hand along prerecorded shapes or paths.

To describe the concept of haptic "playback," consider a simple example. Assume that the goal is to program a haptic device such that it will display a square shaped frame to a passive user. Here, programming of a haptic device as a typical closed-loop control problem is considered. Although this is a well-studied problem in robotics, the following practical issues appear when it is implemented with haptic devices:

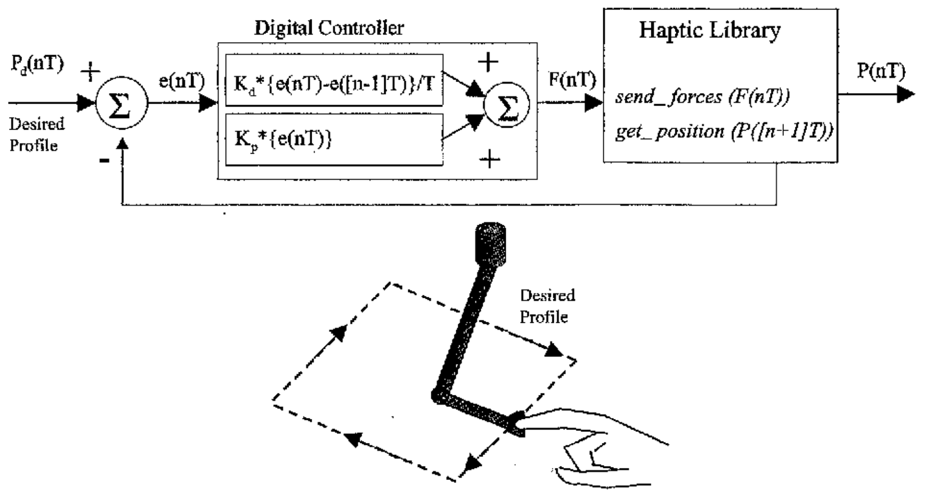


FIG. 6.9. Digital servo loop for commanding the haptic device to play prerecorded haptic stimuli (a frame in this example). P, e, and F represent position, error, and force vectors respectively. $K_d$ and $K_p$ are derivative and proportional control gains.

- Haptic devices are typically "open-loop" systems. The user manipulates the end-effector of the robot to control the end position, but the dynamics of interactions are not governed by the principles of closed-loop control systems. In other words, you need to design your own controller to command your haptic device to play back prerecorded haptic stimuli.
- Unless you build your own haptic device and do not purchase from a manufacturer, you most likely do not know the kinematics (e.g., mass and length of each link) and dynamic properties (e.g., friction, performance of motors) of your device exactly. Haptic devices, sold by a manufacturer, usually come with a library that includes function calls for acquiring position and orientation of the end-effector and for sending force commands to motors (see section 2). Hence, the goal is to use these function calls to control the movements of the haptic device and navigate from one point to another in 3-D space while compensating for positional errors. One can design a controller to achieve this goal (see Fig. 6.9; we use a PD controller to track the desired profile).

It is possible to play back prerecorded forces to the user even when the user is actively interacting with the environment by using, visual cues. Dang, Annaswamy, & Srinivasan (2001) have developed an epidural injection simulator in which a training feature is the capability to display an expert's path and forces to a novice in two ways. The expert's path is displayed visually either as a "tunnel" through which the novice could traverse or as a cursor moving along the prerecorded path that the trainee needs to follow closely to feel the prerecorded forces continuously.

## 7. SHARED HAPTICS: HAPTIC INTERACTION BETWEEN NETWORKED USERS

Another potentially promising area is haptic interaction among a group of users sharing the same VE over a network (see chaps. 16 and 44, this volume). Since the haptic loop requires a high update rate of around 1 kHz for stable interactions, and changes in the visual scene will require frequent update of the haptic database, developing a network protocol that can provide sufficient bandwidth with minimum latency to a group of distributed users is a quite challenging problem. For shared environments to be effective in performing collaborative tasks that involve haptic feedback, several network architectures have been proposed (Buttolo, Oboe, & Hannaford 1997; Wilson, Kline-Schoder, Kenton, & Hogan, 1999), and physics-based models that simulate haptic interactions among users have begun to be developed (Basdogan et al., 2000). Because of network time delays, the difficult nature of some collaborative tasks, and lack of knowledge on user abilities and behaviors, the problem of developing a universal model for shared haptics could be too complex. For example, it is obvious that time delays during the transfer and processing of data may easily result in unstable forces and can be harmful to the user (see chap. 41, this volume). Similarly, participants could follow several different strategies to manipulate virtual objects during the execution of a task if an interaction model is established. For example, one could talk about *sequential* versus *simultaneous* types of haptic manipulations. It is even possible to make one user "stronger" than the other (Basdogan et al., 2000). Since limited knowledge in this area makes it almost impossible to integrate all these features into a single interaction model, the type of model selected to simulate haptic interactions between participants in shared environments depend, at this stage, on the collaborative task itself. For example, a simple physics-based model has been developed to simulate haptic interactions between participants (Basdogan et al., 2000). In this model, each participant manipulates his or her own cursor through a stylus attached to the force feedback device placed next to his or her seat. When the participant manipulates the stylus of the haptic device with his or her dominant hand, the cursor moves in 3-D space so that the manipulated object translates or rotates depending on the task. In our experiments, a spring-damper model ($F = k\Delta p + b\Delta\dot{p}$, where $F$ is the force exerted by the user on the ring that is felt by his or her remote partner, $k$ and $b$ are the spring and the damping coefficients, and $\Delta p$ is the displacement of the participant's cursor) was used to control the impedance of interactions between participants and between the participant's cursor and the ring in the scene (see Fig. 6.10). This model simply

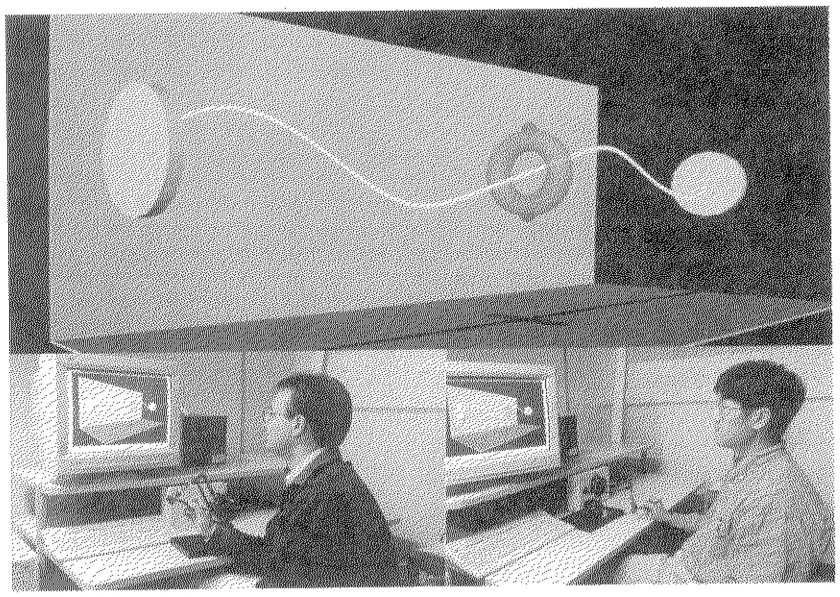


FIG. 6.10. We developed a shared VE setup that enables two people, at remote locations, to interact with each other through visual and haptic displays. An experiment was designed to investigate the role of haptics in collaborative manipulation. In our experiment, participants were asked to hold and move a ring on a wire in a collaborative manner, as depicted in this figure. To eliminate time delays due to network connections, we bifurcated the signals from a single host and displayed it on two separate monitors and haptic interfaces.

simulates the translational movements of the ring on a wire and pulling and pushing forces between the participants. Hence, if a participant pulls or pushes his own cursor the remote partner feels the forces. Visually, however, the ring remains rigid (i.e., no deformation of the ring is displayed graphically). The rotation of the ring due to unbalanced forces applied by the participants was prevented to make the task easier. Moreover, only "simultaneous" haptic interactions were supported such that the ring did not move if both participants did not apply sufficient forces to the ring at the same time.

## 8. SUMMARY AND FUTURE

Computer haptics is concerned with the development of software algorithms that enable a user to touch, feel, and manipulate objects in VEs through a haptic interface. It primarily deals with the computation of forces to be displayed to the user in response to user's actions. The demands placed on it depend on the capabilities and limitations of the interface hardware, computational engine, and user, together with the needs of the task to be accomplished. Current haptic interface technology is mature for net force and torque feedback, as in tool usage in the real world. Consequently, current haptic rendering algorithms have been developed mainly for simulating net forces and torques of interaction and give users the feeling of touching and manipulating objects through a stick or a rigid thimble. Point-based rendering, which views the haptic cursor as a point in computing point-object interaction forces, was developed first and is widely used. This was followed by ray-based rendering, which computes line–object interaction forces and torques. More recently, limited success has been achieved in simulating 3-D object–object interactions in which one of the objects is viewed as a collection of points. Using these haptic interaction paradigms, real-time haptic display of shapes, textures, and friction of rigid and deformable objects has been achieved. Haptic rendering of dynamics of rigid objects, and to a lesser extent, linear dynamics of deformable objects, has also been accomplished. Methods for recording and playing back haptic stimuli, as well as algorithms for haptic interactions between multiple users in shared VEs, are beginning to emerge.

In the future, capabilities of haptic interface devices are expected to improve primarily in two ways: (1) improvements in both desktop and wearable interface devices in terms of factors such as inertia, friction, workspace volume, resolution, force range, and bandwidth; (2) development of tactile displays to simulate direct contact with objects, including temperature patterns. These are expected to result in multifinger, multihand, and even whole body displays, with heterogeneous devices connected across networks. Even with current rapid expansion of the capabilities of affordable computers, the needs of haptic rendering with more complex interface devices will continue to stretch computational resources. Currently, even with point-based rendering, the computational complexity of simulating the nonlinear dynamics of physical contact between an organ and a surgical tool, as well as surrounding tissues is very high. Thus there will be continued demand for efficient algorithms, especially when the haptic display needs to be synchronized with the display of visual, auditory, and other modalities (see chap. 21, this volume). Similar to graphics accelerator cards used today, it is quite likely that repetitive computations will need to be done through specialized electronic hardware, perhaps through parallel processing. Given all the complexity and need for efficiency, in any given application the central question will be how good the simulation needs to be to achieve a desired goal.

## 9. REFERENCES

Adachi, Y., Kumano, T., & Ogino, K. (1995). Intermediate representation for stiff virtual objects. In *Proceedings of the IEEE Virtual Reality Annual International Symposium* (pp. 203–210). Research Triangle Park, NC: IEEE.

Adams, R., & Hannaford, B. (1998). A two-port framework for the design of unconditionally stable haptic interfaces. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Victoria, Canada: IEEE.

Avila, R. S., & Sobierajski, L. M. (1996). A haptic interaction method for volume visualization. In *IEEE Proceedings of Visualization* (pp. 197–204).

Balaniuk, R., & Costa, I. F. (2000). LEM—An approach for physically based soft tissue simulation suitable for haptic interaction. In *Proceedings of the Fifth Phantom Users Group Workshop.*

Baraff, D. (1994). Fast contact force computation for nonpenetrating rigid bodies. *ACM (Proceedings of SIGGRAPH), 28*, 23–34.

Baraff, D. (1997). An introduction to physically based modeling: Rigid body simulation 1—Unconstrained rigid body dynamics. *SIGGRAPH'97 Tutorial Notes.*

Basdogan C. (1999, August). Force reflecting deformable objects for virtual environments. In *Haptics: From basic principles to advanced applications, SIGGRAPH '99 Tutorial Notes, 26th International Conference on Computer Graphics and Interactive Techniques, Course No. 38.*

Basdogan C. (2000). Course name: Simulating minimally invasive surgical procedures in virtual environments: From tissue mechanics to simulation and training. Medicine Meets Virtual Reality (MMVR '2000), Jan. 27–30, Irvine, CA, Available: http://www.amainc.com/MMVR/MMVR2000pro.html, http://eis.jpl.nasa.gov/~ basdogan

Basdogan, C. (2001). Real-time simulation of dynamically deformable finite element models using modal analysis and spectral lanczos decomposition methods. In *Medicine Meets Virtual Reality*, pp. 16–52.

Basdogan, C., Ho, C., & Srinivasan, M. A. (1997). A ray-based haptic rendering technique for displaying shape and texture of 3D objects in virtual environments. *ASME Winter Annual Meeting* (pp. 61, 77–84). Dallas, TX: ASME.

Basdogan, C., Ho, C., & Srinivasan, M. A. (2001). Virtual environments for medical training: Graphical and haptic simulation of common bile duct exploration. *IEEE/ASME Transactions on Mechatronics, 6*(3), 267–285.

Basdogan, C., Ho, C., Srinivasan, M.A., & Slater, M. (2000). An experimental study on the role of touch in shared virtual environments, In *ACM Human Computer Interactions*, Vol. 7, No. 4, pp. 440–463.

Basdogan C., Ho, C., Srinivasan, M. A., Small, S., & Dawson, S. (1998). Force interactions in laparoscopic simulations: Haptic rendering of soft tissues. *Proceedings of the Medicine Meets Virtual Reality VI Conference* (pp. 385–391).

Bathe, K. (1996). *Finite Element Procedures.* NJ: Prentice Hall.

Bier, E. A., & Sloan, K. R. (1986). Two-part texture mapping. *IEEE Computer Graphics and Applications*, 40–53.

Blinn, J. F. (1978). Simulation of wrinkled surfaces. *ACM (Proceedings of SIGGRAPH), 12*(3), 286–292.

Buttolo, P. (1996). *Characterization of human pen grasp with haptic displays.* Unpublished Ph.D. Dissertation, University of Washington, Seattle, WA.

Buttolo, P., Oboe, R., & Hannaford, B. (1997). Architectures for shared haptic virtual environments. *Computers and Graphics, 21*, 421–429.

Cohen, J., Lin, M., Manocha, D., & Ponamgi, K. (1995). I-COLLIDE: An interactive and exact collision detection system for large-scaled environments. *Proceedings of ACM Interactive 3D Graphics Conference* (189–196).

Colgate, J. E., & Brown, J. M. (1994). Factors affecting the z-width of a haptic display. *Proceedings of IEEE International Conference on Robotics and Automation* (pp. 3205–3210).

Cotin, S., Delingette, H., & Ayache, N. (1999). Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions on Visualization and Computer Graphics, 5*(1), 62–73.

Cover S. A., Ezquerra N. F., O'Brien J., Rowe R., Gadacz T., & Palm E. (1993). Interactively deformable models for surgery simulation. *IEEE Computer Graphic and Applications*, November, 65–78.

Dachille, F., Qin, H., Kaufman, A., & El-sana, J. (1999). Haptic sculpting of dynamic surfaces. In *ACM Symposium on Interactive 3D Graphics.*

Dang, T., Annaswamy, T. M., & Srinivasan, M. A. (in press). Development and evaluation of an epidural injection simulator with force feedback for medical training. In *Medicine Meets Virtual Reality.*

De, S., Kim, J., & Srinivasan, M. A. (2001). A meshless numerical technique for physically based real-time medical simulations. In *Medicine Meets Virtual Reality*, pp. 113–118.

De, S., & Srinivasan, M. A. (1999). Thin-walled models for haptic and graphical rendering of soft tissues in surgical simulations. In *Medicine Meets Virtual Reality* (Eds: Westwood, et al.), IOS Press.

Ebert, D. S., Musgrave F. K., Peachey D., Perlin K., & Worley S. (1994). *Texturing and Modeling.* Cambridge, MA: AP Professional.

Edwards, J., & Luecke, G. (1996). Physically based models for use in a force feedback virtual environment. *Japan/USA Symposium on Flexible Automation* (pp. 221–228). ASME.

Ellis, R. E., Sarkar, N., & Jenkins, M. A. (1996). Numerical methods for the haptic presentation of contact: Theory, simulations, and experiments. *Proceedings of the ASME Dynamic Systems and Control Division* (DSC-Vol. 58, pp. 413–420).

Foley, J. D., van Dam, A., Feiner, S. K., & Hughes, J. F. (1995). *Computer graphics: Principles and practice.* Addison-Wesley.

Fritz & Barner (1996). Haptic scientific visualization. In J. K. Salisbury & M. A. Srinivasan (Eds.), *Proceedings of the First PHANToM Users Group Workshop.* MIT-AI TR-1596 and RLE TR-612.

Fukui, Y. (1996). Bump mapping for a force display. *Proceedings of the First PHANToM User Group Workshop* [online]. Available: http://www.ai.mit.edu/conferences/pug/pug-proceedings.html

Gottschalk, S., Lin, M., & Manocha, D. (1996). OBB-Tree: A hierarchical Structure for Rapid Interference Detection. *ACM (Proceedings of SIGGRAPH).*

Green, D. F., & Salisbury, J. K. (1997). Texture sensing and simulation using the PHANToM: towards remote sensing of soil properties. In *Proceedings of the Second PHANToM Users Group Workshop.*

Ho, C., Basdogan, C., & Srinivasan, M. A. (1999). An efficient haptic rendering technique for displaying 3D polyhedral objects and their surface details in virtual environments. *Presence: Teleoperators and Virtual Environments, 8*(5), 477–491.

Ho, C., Basdogan, C., & Srinivasan, M. A. (2000). Modeling of force and torque interactions between a line segment and triangular surfaces for haptic display of 3D convex objects in virtual and teleoperated environments [Special issue] *International Journal of Robotics, 19*(7), 668–684.

Hollerbach J., & Johnson, D. (in press). In *Human and Machine Haptics* (Eds: Srinivasan, Cutkosky, Howe, and Salisbury).

Hubbard, P. (1995). Collision detection for interactive graphics applications. *IEEE Transactions on Visualization and Computer Graphics, 1*(3), 219–230.

Hsu, W. M., Hughes, J. F., & Kaufman, H. (1992). Direct manipulation of free-form deformations. *Computer Graphics (Proceedings of the SIGGRAPH), 26*(2), 177–184.

James, D., & Pai, D. (1999). ARTDEFO: Accurate real time deformable objects, In *Computer Graphics (SIGGRAPH '99 Conference Proceedings).*

Krebs, H. I., Hogan, N., Aisen, M. L., & Volpe, B. T. (1998). Robot-aided neurorehabilitation. *IEEE Transactions on Rehabilitation Engineering, 6*(1), 75–86.

Latimer, C. (1997). *Haptic interaction with rigid objects using real-time dynamic simulation.* Unpublished master's thesis, Massachusetts Institute of Technology.

Lee, Y., Terzopoulos, D., & Waters, K. (1995). Realistic modeling for facial animation. *Proceedings of the SIGGRAPH,* 55–62.

Lin, M. (1993). *Efficient collision detection for animation and robotics.* Unpublished doctoral dissertation, University of California, Berkeley.

MacLean, K. (1996). The "haptic camera": A technique for characterizing and playing back haptic properties of real environments. *Proceedings of ASME Dynamic Systems and Control Division* (DSC-Vol. 58, pp. 459–467).

Mandelbrot, B. (1982). *The Fractal Geometry of Nature.* San Francisco; W. H. Freeman.

Mark, W., Randolph S., Finch, M., Van Verth, J., & Taylor, R.M. (1996). Adding force feedback to graphics systems: issues and solutions. *Computer Graphics: Proceedings of SIGGRAPH '96,* 447–452.

Max, N. L., & Becker, B. G. (1994). Bump shading for volume textures. *IEEE Computer Graphics and Applications, 4,* 18–20.

McNeely, W. A., Puterbaugh K. D., and Troy, J. J. (1999). Six degree-of-freedom haptic rendering using voxel sampling. *Proceedings of SIGGRAPH,* 401–408.

Minsky, M., Ming, O., Steele, F., Brook, F. P., & Behensky, M. (1990). Feeling and seeing: issues in force display. *Proceedings of the Symposium on 3D Real-Time Interactive Graphics, 24,* 235–243.

Mirtich, B. (1996). *Impulse-based dynamic simulation of rigid body systems.* Unpublished doctoral dissertation, University of California, Berkeley.

Mirtich, B., & Canny, J. (1995). Impulse-based simulation of rigid bodies. *Proceedings of Symposium on Interactive 3D Graphics,* April.

Morgenbesser, H. B., & Srinivasan, M. A. (1996). Force shading for haptic shape perception. *Proceedings of the ASME Dynamic Systems and Control Division, 58,* 407–412.

Ng, H., & Grimsdale, R. (1996). Computer Graphics Techniques for Modeling Cloth. *IEEE Computer Graphic and Applications,* 28–41.

Oboe, R., & Fiorini, P. (1998). A design and control environment for internet-based telerobotics. *The International Journal of Robotics Research, 17*(4), 433–449.

Perlin, K. (1985). An image synthesizer. *ACM SIGGRAPH, 19*(3), 287–296.

Ruspini, D. C., Kolarov, K., & Khatib, O. (1996). Robust haptic display of graphical environments. In J. K. Salisbury & M. A. Srinivasan (Eds.), *Proceedings of the First PHANToM Users Group Workshop,* MIT-AI TR-1596 and RLE TR-612.

Ruspini, D. C., Kolarov, K., & Khatib, O. (1997). The haptic display of complex graphical environments. *ACM (Proceedings of SIGGRAPH),* 345–352.

Salcudean, S. E., & Vlaar, T. D. (1994). On the emulation of stiff walls and static friction with a magnetically levitated input/output device. *Proceedings of the ASME* DSC, *55*(1), 303–309.

Salisbury, J. K., Brock, D., Massie, T., Swarup, N., & Zilles, C. (1995). Haptic rendering: Programming touch interaction with virtual objects. In *Proceedings of the ACM Symposium on Interactive 3D Graphics.* Monterey, CA.

Salisbury, J. K., & Srinivasan, M. A. (1997). Phantom-based haptic interaction with virtual objects. *IEEE Computer Graphics and Applications, 17*(5).

Salisbury, J. K., & Tarr, C. (1997). Haptic rendering of surfaces defined by implicit functions. *Proceedings of the ASME* (DSC-Vol. 61, 61–67).

Sederberg, T. W., & Parry S. R. (1986). Free-form deformation of solid geometric models. *SIGGRAPH Proceedings on Computer Graphics, 20*(4), 151–160.

Siira, J., & Pai, D. K. (1996). Haptic texturing—A stochastic approach. *Proceedings of the IEEE International Conference on Robotics and Automation* (pp. 557–562). Minneapolis, MN: IEEE.

Srinivasan M. A. (1989). Surface deflection of primate fingertip under line load. *Journal of Biomechanics,* 22(4), 343–349.

Srinivasan, M. A. & Basdogan, C. (1997). Haptics in virtual environments: Taxonomy, research status, and challenges. *Computers and Graphics, 21*(4), 393–404.

Srinivasan M. A., Beauregard G. L., & Brock, D. L. (1996). The impact of visual information on haptic perception of stiffness in virtual environments. *Proceedings of the ASME Dynamic Systems and Control Division,* (DSC-Vol. 58, pp. 555–559).

Srinivasan M. A., Whitehouse J. M., & LaMotte, R. H. (1990). Tactile detection of slip: Surface microgeometry and peripheral neural codes. *Journal of Neurophysiology, 63*(6), 1323–1332.

Stewart, P., Chen, Y., & Buttolo, P. (1997). Direct integration of haptic user interface in CAD systems. *Proceedings of ASME* (DSC-Vol. 61, pp. 93–99).

Swarup, N. (1995). *Haptic interaction with deformable objects using real-time dynamic simulation.* Unpublished master's thesis, Massachusetts Institute of Technology.

Thompson, T. V., Johnson, D. E., & Cohen, E. (1997). Direct haptic rendering of sculptured models. *Proceedings of the Symposium on Interactive 3D Graphics* (pp. 1–10). Providence, RI:

Turk, G. (1991). Generating textures on arbitrary surfaces using reaction-diffusion. *ACM (Proceedings of SIGGRAPH), 25*(4), 289–298.

Watt, A., & Watt, M. (1992). *Advanced Animation and Rendering Techniques.* New York: Addison-Wesley.

Wilson J., Kline-Schoder, Kenton, M. A., & Hogan, N. (1999). Algorithms for network-based force feedback. In J. K. Salisbury & M. A. Srinivasan (Eds.), *Proceedings of the Fourth PHANToM Users Group Workshop,* (AI Lab Tech. Rep. No. 1675 and RLE Tech. Rep. No. 633). Cambridge, MA: Massachusetts Institute of Technology.

Wijk, J. J. V. (1991). Spot noise. *ACM (Proceedings of SIGGRAPH), 25*(4), 309–318.

Witkin A., Barraff D., & Kass, M. (1998). Tutorial notes on *"An Introduction to Physically-Based Modeling",* SIGGRAPH '98.

Witkin, A., & Kass, M. (1991). Reaction-diffusion textures. *ACM (Proceedings of SIGGRAPH), 25*(4), 299–308.

Worley, S. (1996). A cellular texture basis function. *ACM (Proceedings of SIGGRAPH),* 291–294.

Zilles, C. B., & Salisbury, J. K. (1995). A constraint-based god–object method for haptic display. *IEEE International Conference on Intelligent Robots and System, Human Robot Interaction, and Co-operative Robots, 3,* 146–151.