Haptic Rendering in Virtual Environments

Cagatay Basdogan[1, 2], Mandayam A. Srinivasan[2]

[1]*Jet Propulsion Laboratory*
California Institute of Technology
Pasadena, CA 91101
Cagatay.Basdogan@jpl.nasa.gov

[2]*Laboratory for Human and Machine Haptics*
Massachusetts Institute of Technology
Cambridge, MA 02139
basdogan, 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 (ref. **Biggs and Srinivasan Chapter**), it is now possible to simulate the feel of even fine surface textures on rigid complex shapes under dynamic conditions. Starting from the early nineties, significant progress has occurred in our ability to model and simulate haptic interactions with 3D virtual objects in real-time (Salisbury and Srinivasan, 1997; Srinivasan and Basdogan, 1997). The rapid increase in the number workshops, conference sessions, community web pages, and electronic journals on haptic displays and rendering techniques[1] indicates growing interest in this exciting new area of research, which we call *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. We have already seen various applications of computer haptics in the areas of medicine (surgical simulation, tele-medicine, haptic user interfaces for blind people, rehabilitation of patients with neurological disorders) entertainment (3D painting, character animation, morphing and sculpting), mechanical design (path planning and assembly sequencing), and scientific visualization (geophysical data analysis, molecular manipulation). We anticipate that more applications are on the way as the devices and rendering techniques improve and the computational power increases. This chapter will primarily focus on the fundamental concepts of haptic rendering with some discussion of implementation details. Although, it is impossible to cite all the relevant work within the constraints of this chapter, we have made attempts to cover the major references. Given that the 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), we restrict ourselves to techniques concerning the former. In general, the concepts discussed in the chapter include: (1)

[1] 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 haptics-e journal at http://www.haptics-e.org/.

haptic interaction paradigms that define the nature of the "haptic cursor" and its interaction with virtual objects and (2) *object property display algorithms* that enable us to render 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 Figure 1). As the user manipulates the probe of the haptic device, the new position and orientation of the haptic probe are acquired, collisions with the virtual objects are detected (i.e. *collision detection*). If a collision is detected, the interaction forces are computed using preprogrammed rules for *collision response*, and conveyed to the user through the haptic device to provide him/her with the tactual representation of 3D 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 &position);        // position and/or orientation of the end-effector
    calculate_force (Vector &force);        // user-defined function to calculate forces
    send_force (Vector force);              // calculate joint torques and reflect forces back to the user
…
```

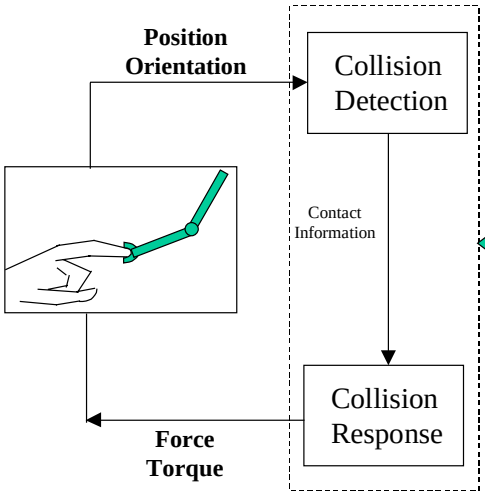
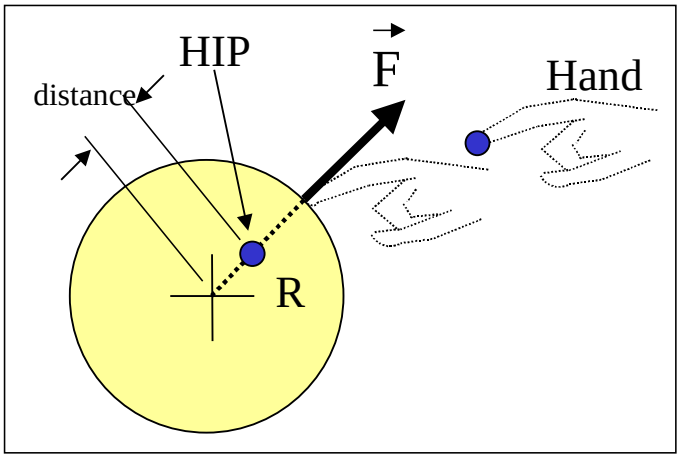


Figure 1. A haptic interaction algorithm is typically made of two parts: (a) collision detection and (b) collision response. The haptic loop seen in the figure requires an update rate of around 1 kHz for stable force interactions. Computationally fast collision detection and response techniques are required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

To describe the basic concepts of haptic rendering, let us consider a simple example: haptic rendering of a 3D frictionless sphere, located at the origin of a 3D virtual space (see Figure 2). Let us 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 we freely explore the 3D space with the haptic probe, the haptic device will be *passive* and will not reflect any force to the user until a contact occurs. Since our 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 our hand to resist further penetration. We can easily compute the magnitude of the reaction force 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 Figure 2.




```
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);

        }

    }
```

Figure 2. Haptic rendering of a 3D 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 located at the origin.

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. While keeping the stiffness coefficient low would make the surface perceived soft, setting a high value would make the interactions unstable by causing undesirable vibrations. Figure 3 depicts the changes in force profile with respect to position for real and virtual walls. Since the position of the probe tip is sampled digitally with 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 and Brown, 1994 and Ellis et al., 1996).

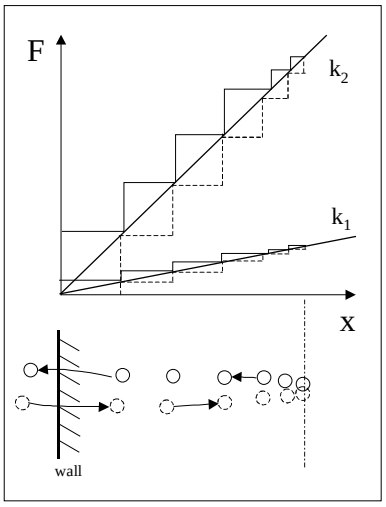


Figure 3. Force-displacement curves for touch interactions with real and virtual walls. In the case of real wall, the force-displacement curve is continuous. However, we see "staircase" effect when simulating touch interactions with a virtual wall. This is due to the fact that haptic device can only 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 stiffness coefficient generates a soft virtual wall, which is not desirable either.

Although the basic recipe for haptic rendering of virtual objects seems easy to follow, rendering of complex 3D 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 us very little CPU time for computing the collisions and reflecting the 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 real world), which is far from what our hands are capable of in real world. However, several haptic rendering techniques have been developed recently to simulate complex touch interactions in virtual environments (reviewed in Salisbury and Srinivasan, 1997; Srinivasan and Basdogan, 1997; Hollerbach and Johnson, 2001). The existing techniques for haptic rendering with force display can be distinguished based on the way the probing object is modeled: (1) a point (Zilles and Salisbury, 1995; Adachi et al., 1995; Avila and Sobierajski, 1996; Ruspini et al., 1997; Ho et al., 1999), (2) a line segment (Basdogan, et al., 1997; Ho et al., 2000), or (3) a 3D object made of 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, also known as the haptic interface point (HIP), interacts with virtual objects (Figure 4b). Each time the 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 Figure 5). For exploring the shape and surface properties of objects in VEs, point-based methods are probably sufficient and could provide the users with similar force feedback as what they would feel when exploring the objects in real environments with the tip of a stick. Using a point-based rendering technique, polyhedrons (Zilles and Salisbury, 1995; Ruspini et al., 1997; Ho et al., 1999), NURBS (Thompson et al., 1997; Stewart et al., 1997), implicit surfaces (Salisbury and Tarr, 1997), and volumetric objects (Avila and 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 the collisions are checked between the finite line and the 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 the point-based approaches (see Figure 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 3D cursor and 3D objects. Also, if the geometric model of the probing object can be simplified to a set of connected line segments, then ray-based rendering technique can be used and will be faster than simulation of full 3D object interactions. For example, we have successfully simulated the haptic interactions between a mechanical shaft and an engine block (Ho et al., 2000) as well as the interactions between laparoscopic surgical instruments and deformable objects (Basdogan, *et al.* 1998; Basdogan et al., 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 six degree of freedom (dof) object-object interactions has to be considered.

The simulation of haptic interactions between two 3D polyhedra is desirable for many applications, but this is computationally more expensive than the point-based and ray-based interactions (see Figure 4d). Although a single point is not sufficient for simulating the force and torque interactions between two 3D virtual objects, a group of points, distributed over the surface of probing object, has been shown to be a feasible solution. For example, McNeely et al. (1999) simulated the touch interactions between 3D model of a teapot and a mechanical assembly.

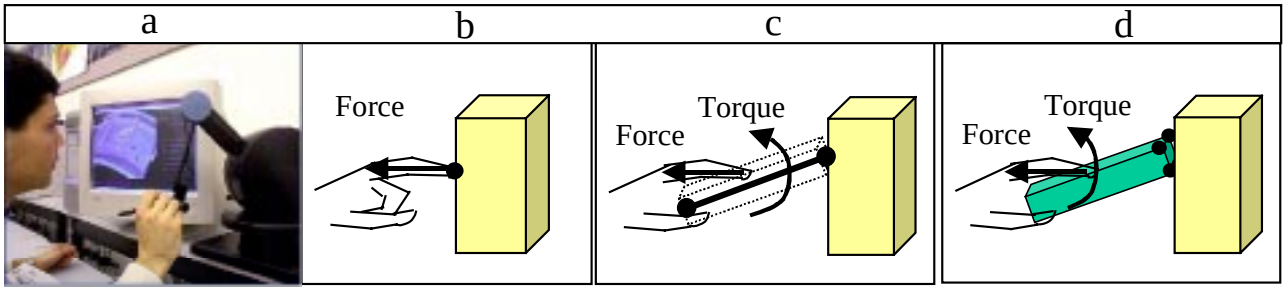


Figure 4. The existing interaction methods for haptic rendering with force display can be distinguished based on the way the probing object is modeled: b) a point, c) a line segment, and d) a 3D object.

Irrespective of the interaction method used, a haptic rendering algorithm must include both collision detection and collision response computations (see Figure 1). Although collision detection has been extensively studied in computer graphics (Lin, 1993; Hubbard, 1995; Gottschalk et al., 1996; Cohen et al., 1995), the existing algorithms are not designed to work directly with haptic devices to simulate touch interactions in virtual environments. Moreover, merely detecting collisions between 3D objects is not enough for simulating haptic interactions. How the collision occurs and how it evolves over time (i.e. *contact history*) are crucial factors (see the discussion in Figure 5 on contact history) in haptic rendering to accurately compute the interaction forces that will be reflected to the user through the haptic device (Ho et al., 1999; 2000).

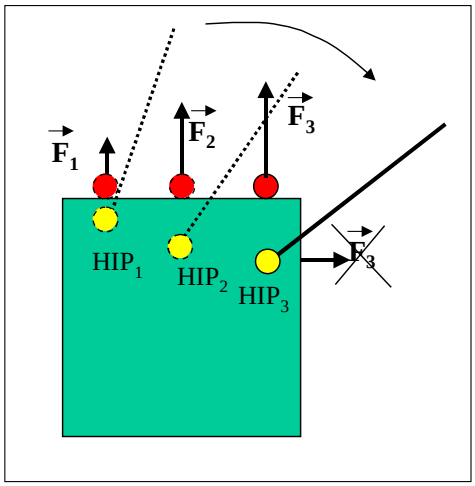


Figure 5. A haptic rendering algorithm that is based on Point interactions must calculate the IHIP for a given HIP at each rendering cycle. IHIP, (also known as the god-object, surface contact point, or proxy point), can be considered as the surface point of the object in which the tool or finger would be in contact. The computation of this point relies on the contact history. In the figure, for example, ignoring contact history and always choosing the closest point on the object surface as our 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.

Although the 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 (a) space partitioning techniques (i.e. partitioning the space that encloses the object into smaller sub-spaces in advance for faster detection of first contact), (b) local search approaches (i.e. searching only the neighboring primitives for possible new contacts), and (c) 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 multi-threading 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 (Ellis et al., 1996; Basdogan, 2000).

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 3D object (Foley et al., 1995). Then, shading algorithms are applied to shade each polygon by interpolating the light intensities (Gourand 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 and Watt, 1992). One can integrate a similar approach into the study of haptics to convey the feel of haptically smooth object surfaces (Morgenbesser and Srinivasan, 1996, Fukui, 1996; Basdogan et al, 1997; Ruspini et al, 1997). By using *force-shading* technique suggested by Morgenbesser and Srinivasan (1996), we can reduce the force discontinuities and make the edges of polyhedral object feel smoother. To implement the force shading with polygonal surfaces (Basdogan et al., 1997), we pre-compute the surface normal at each vertex by averaging the surface normals of neighboring polygons, weighted by their neighboring angles. During the real-time computations, we first detect the collision point that divides the contacted triangle into three sub-triangles. 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 sub-triangles:

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

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 and virtual surfaces without *friction* feel "icy-smooth" when they are explored with a haptic device. Similarly, most of the surfaces in nature are covered with some type of *texture* that is sensed and distinguished quite well by our tactile system (Srinivasan et al, 1990). Haptic texture is a combination of small-scale variations in surface geometry and its adhesive and frictional characteristics. However, displaying the 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 the friction and the 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 Figure 6).

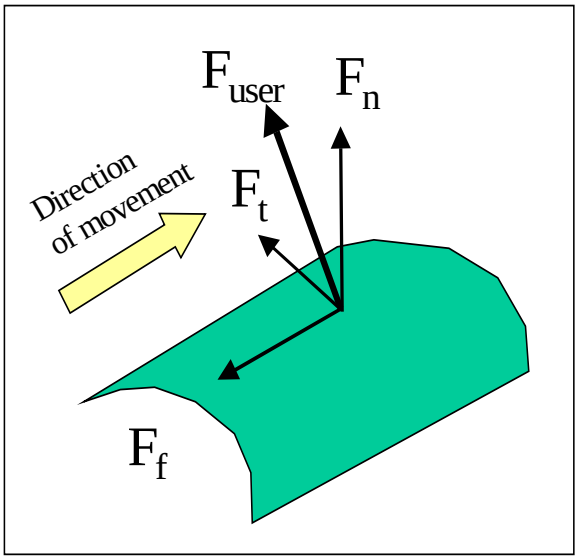


Figure 6. Forces acting on the user ($F_{user} = F_n + F_t + F_f$) during the haptic simulation of friction and textures. The normal force can be computed using a simple physics-based model such as Hooke's law ($F_n = k\,\Delta x$, where $|\Delta x|$ is the depth of penetration of the haptic probe into the virtual surface). To simulate coulomb friction, we need to create a force ($F_f = \mu F_n$, where μ is the coefficient of friction) that is opposite to the direction of the movement. To simulate textures, we change the magnitude and direction of the normal force vector using the gradient vector of the texture field at the contact point.

Salcudean and Vlaar (1994) and Salisbury et al. (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 object surface. By changing the mean value of friction coefficient and its variation, more sophisticated frictional surfaces such as periodic ones (Ho et al., 1999) and various grades of sandpaper (Green and Salisbury, 1997) can be simulated as well.

Haptic perception and display of textures in virtual environments require a thorough investigation, primarily because the textures in nature come in various forms. Luckily, graphics texturing has been studied extensively and we 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 non-smooth surface by perturbing the surface normals. Blinn (1978) initially proposed this technique as a method of graphically rendering 3D 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 the 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 ($\bar{N}_s$) to generate bumpy or textured surfaces that can be sensed tactually by the user in virtual environments (Basdogan et al., 1997). The perturbed surface normals ($\bar{M}$) can be computed using the following formulation:

$$\vec{M} = \vec{N}_s - \nabla h + (\nabla h . \vec{N}_s)\vec{N}_s$$

$$\nabla h = \frac{\partial h}{\partial x}\hat{i} + \frac{\partial h}{\partial y}\hat{j} + \frac{\partial h}{\partial z}\hat{k}$$

where, h(x,y,z) represents the texture field function and ∇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 et al. (1990). The formulation given above extends this concept to 3D surfaces.

In general, the haptic texturing techniques can be classified into two parts: (a) image-based and (b) procedural.

Image-based haptic texturing: This class of haptic texturing deals with constructing a texture field from a 2D image data. In computer graphics, digital images are wrapped around 3D objects to make them look more realistic. While a graphical texture consists of 2D 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 and Sloan, 1986; Watt and Watt, 1992) can be followed. The first stage in this technique is to map the 2D image to a simple intermediate surface such as 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).

Procedural haptic texturing: The goal of procedural haptic texturing is to generate synthetic textures using mathematical functions. The function usually takes the coordinate (x,y,z) as the input and returns the height value and its gradient as the outputs. For example, several investigators have implemented the well-known noise texture (Perlin, 1985; Ebert et al. 1994) to generate stochastic haptic textures (Siira and Pai, 1996;

Fritz and Barner, 1996; Basdogan et al., 1997). Fractals are also appropriate for modeling natural textures since many objects seem to exhibit self-similarity (Mandelbrot, 1982). We have used the fractal concept in combination with the 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, we (Ho et al., 1999) have implemented haptic versions of reaction-diffusion textures (Turk, 1991, Witkin and Kass, 1991), the 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, the deformable objects have gained a new characteristic. Now, our deformable models need to estimate not only the direction and the amount of deformation of each node of 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 the deformation techniques is according to the approach followed by the researchers to deform the surfaces: *geometry-* or *physics-based* deformations. In geometry-based deformations, the object or the surrounding space is deformed, based purely on geometric manipulations. In general, the user manipulates vertices or control points that surround the 3D 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 sub-grouped as follows (see Basdogan, 1999 and 2000 for more details):

A. 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 the Hooke's law, where the 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, the polynomial functions that fit experimental data on fingerpad deformation (Srinivasan, 1989) were used to remap the vertices of other organs (Basdogan, et al., 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 et al. (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) and computer-aided design (CAD) and haptic sculpting (Edwards and Luecke, 1996;

Dachille et al., 1999) can be found in the literature. Here, the key advantages of using force feedback are to increase intuition, to control deformations, and to support the implementation of various constraints.

B Physics-based Deformation Models

In comparison to geometry-based models, the 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 consists 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 et al., 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 et al., 1995; Ng and Grimsdale, 1996). Swarup (1995) demonstrated the implementation of this technique to haptic simulation of deformable objects.
- *Finite Element based:* The volume occupied by 3D object is divided into finite elements, properties of each element is formulated and the elements are assembled together to study the 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 FEM with haptic displays (see Cotin et al., 1999 and James and Pai, 1999 for static computations, 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 3D problem into an effectively 2D one (see also Belaniuk and Costa, 2000). In Basdogan, et al, (2001), to compute the dynamical deformations and the interaction forces, we implemented a modal analysis approach such that only the most significant vibration modes of the object were selected. . In this paper, we have also incorporated interactions between two deformable objects, one finite element based and the other particle based. Basdogan (2001) has 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, et al., 2001).

In our earlier studies, we have proposed a loose coupling between the force and displacement computations to take advantage of the human perceptual limitations (Basdogan et al., 1998). In this approach, we used the vertex-based or spline-based approaches to generate smooth visual deformations, while the interaction forces were computed and reflected to the user based on a simple spring and a damper model between the new and the home positions of the contact point. Since the human perception of position and motion is dominantly influenced by visual cues in interacting with objects (Srinivasan, et al, 1996), the loose coupling between the force and displacement computations was not readily sensed by the users during our simulations (see Model A in Figure 7). However, this architecture is limited to the simulation of geometric-based deformations (e.g.

sculpting and animation). Since force and displacement computations are 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 Figure 7). In this architecture, forces and displacements computed in the previous cycle are continuously supplied back to the physics-based model to generate the new values in the next cycle.

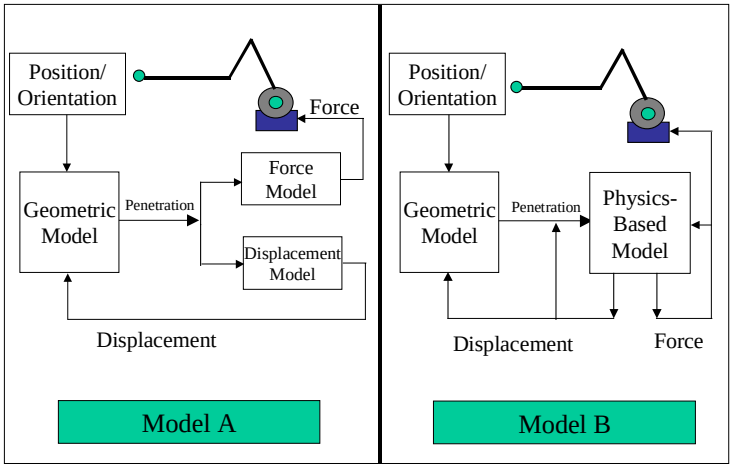


Figure 7. The computational architectures for simulating force-reflecting deformable objects in VEs. In model A, the force and displacement computations are loosely coupled. In model B, the physics-based model generates forces and displacements simultaneously, which are then fed back to the model to generate the new values. Although this architecture is computationally expensive to implement, it is more comprehensive than the architecture A (Basdogan, 1999).

5. Rendering of Dynamic Objects

Simulating haptic interactions with 3D objects that translate and rotate over time is another challenge, which becomes computationally quite intensive if the 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 Figure 8). Techniques to calculate forces and torques based on the principles of rigid body dynamics (Baraff, 1994) and impulse mechanics (Mirtich and Canny, 1995 and Mirtich, 1996) have been developed for the graphical simulation of floating multiple objects. Latimer (1997) discusses the extension of these techniques to haptics domain while Colgate and Brown (1994) and Adams and Hannaford (1998) address the stability issues.

To describe the basic concepts of haptic interaction with dynamic objects, let us consider an example (see Figure 8a). Let us assume that the 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 the equations that govern the dynamics of rigid body motion. These equations can be solved in real-time using a numerical integration method, such as Euler integration, to update the state of the object at each cycle of the haptic loop:

$$\vec{a}_{t+\Delta t} = M^{-1}.\vec{F}_{total}$$
$$\vec{v}_{t+\Delta t} = \vec{v}_t + \Delta t\, \vec{a}_{t+\Delta t}$$
$$\vec{p}_{t+\Delta t} = \vec{p}_t + \Delta t\, \vec{v}_{t+\Delta t}$$

$$\vec{\alpha}_{t+\Delta t} = I^{-1}.\vec{T}_{total}$$
$$\vec{\omega}_{t+\Delta t} = \vec{\omega}_t + \Delta t\, \vec{\alpha}_{t+\Delta t}$$
$$\vec{\theta}_{t+\Delta t} = \vec{\theta}_t + \Delta t\, \vec{\omega}_{t+\Delta t}$$

where, $\vec{F}_{total}$ and $\vec{T}_{total}$ represent the total force and torque acting along a line passing through the center of mass of the object, $\vec{a}, \vec{v}, \vec{p}$ and $\vec{\alpha}, \vec{\omega}, \vec{\theta}$ represent the linear and angular acceleration, velocity and position vectors. In addition, M and I are the mass and inertia matrices of the object and Δt is the sampling time (e.g. Δt = 0.001 seconds if the haptic loop is updated at 1 kHz). In this formulation, the linear quantities can be computed either in object or global coordinate frames whereas the angular quantities need to be computed only in the object coordinate frame. When a force is applied to the object, if there are no constraints, the object will move to a new position defined by $\vec{p}_{t+\Delta t}$ vector while rotating by a total amount of $|\vec{\theta}_{t+\Delta t}|$ degrees. The total force acting on the object ($\vec{F}_{total}$) 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 $\vec{F}_{total}$ vectors (see Figure 8). The details on computation of mass and inertia matrices for polygonal objects 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 Figure 8).

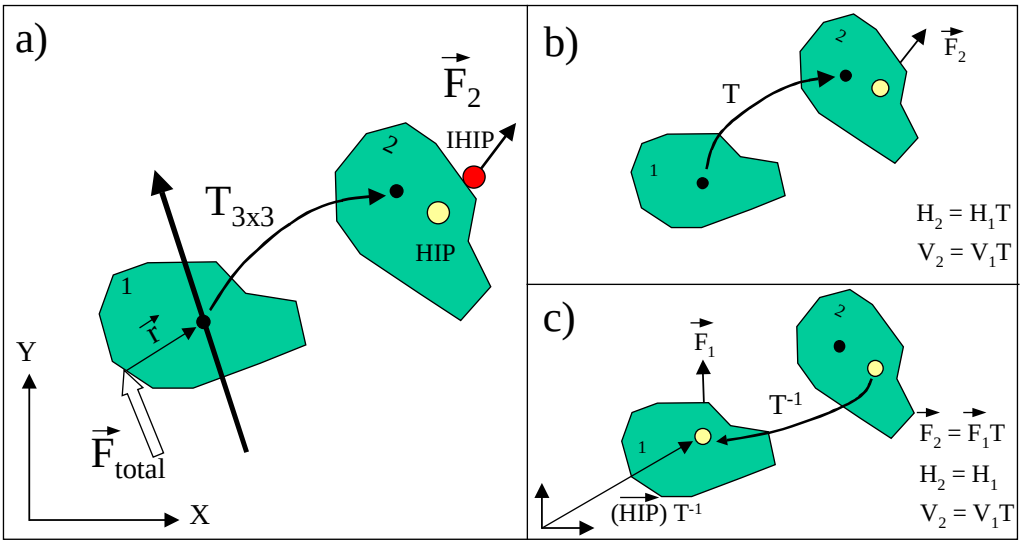


Figure 8. Modeling point-based haptic interactions with a floating rigid object in virtual environments. In the figure, T is the transformation matrix, $\vec{F}_2$ represents the force acting on the user through a haptic display, H and V represent the haptic and visual coordinates of the object respectively. Here, we discuss two different methods for computing the interaction forces when the object is transformed from state 1 to state 2. 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 the visual coordinates are updated. First, HIP is multiplied with the inverse of the transformation matrix and the 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 ($\vec{F}_2 = \vec{F}_1.T$) 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) since the haptic coordinates of the object are not required to be updated at each iteration when the object is transformed to a new state.

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 pre-recorded 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 your screen). In another application, pre-programmed haptic devices can be used in neuro-rehabilitation to improve the motor-control skills of patients with neurological disorders by displaying pre-recorded haptic trajectories (Krebs et al., 1998). In both of these examples, the user will be a passive participant of the system though he/she may have some control over the device. As the user holds the end-effector of the device gently, the device guides his/her hand along pre-recorded shapes or paths.

To describe the concept of haptic “play-back”, let us consider a simple example. Let us assume that our goal is to program a haptic device such that it will display a square shaped frame to a passive user. Here, we can

consider the programming of a haptic device as a typical closed-loop control problem Although this is a well-studied problem in robotics, the following practical issues appear when it is implemented with haptic devices:

- 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 pre-recorded 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 the motors (see Section 2 of this chapter). Hence, your goal is to use these function calls to control the movements of your haptic device and navigate from one point to another in 3D space while compensating for positional errors. One can design a controller to achieve this goal (see Figure 9; we use a PD controller to track the desired profile).

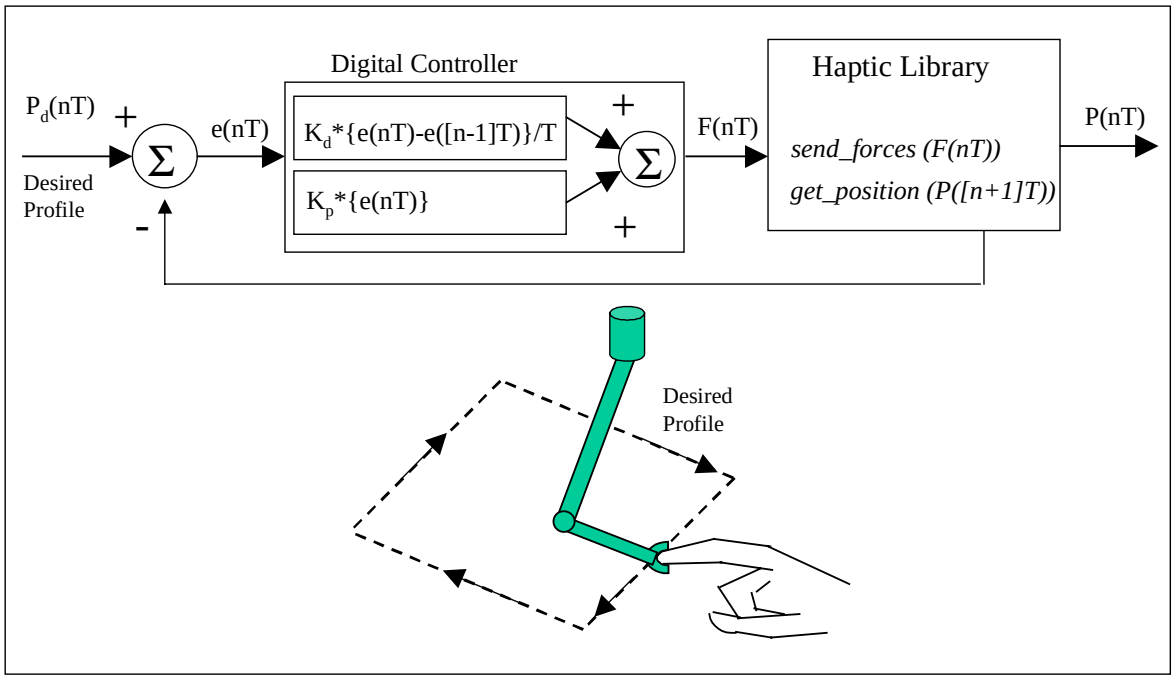


Figure 9. The digital servo loop for commanding the haptic device to play pre-recorded haptic stimuli (a frame in this example). P, e, F represent position, error, and force vectors respectively. K_d and K_p are derivative and proportional control gains.

It is however possible to play back pre-recorded forces to the user even when the user is actively interacting with the environment by using visual cues. Dang, et al (2001) have developed an epidural injection simulator in which a training feature is the capability to display an expert's path and forces to the 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 which the trainee needs to follow closely to feel the pre-recorded 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. 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 et al., 1997; Wilsonet al., 1999) and physics-based models that simulate the haptic interactions among the users have begun to be developed (Basdogan et al., 2001). Because of network time delays, the difficult nature of some collaborative tasks, and our 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. Similarly, participants could follow several different strategies to manipulate the virtual objects during the execution of the task if an interaction model is established. For example, we could talk about '*sequential*' versus '*simultaneous'* types of haptic manipulations. It is even possible to make one user '*stronger*' than the other (see Basdogan, 2000). Since our 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, we have developed a simple physics-based model to simulate the haptic interactions between participants (Basdogan et al, 2000). In our model, each subject manipulates his/her own cursor through a stylus attached to the force feedback device placed next to his/her seat. When the subject manipulates the stylus of the haptic device with his/her dominant hand, the cursor moves in 3D 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/her remote partner, k and b are the spring and the damping coefficients, and Δp is the displacement of the subject's cursor) was used to control the impedance of interactions between the participants and between the subject's cursor and the ring in the scene (see Figure 10). This model simply simulates the translational movements of the ring on a wire and pulling and pushing forces between the subjects. Hence, if a subject pulls or pushes his own cursor, the remote partner feels the forces. Visually, however the ring remained rigid (i.e. no deformation of the ring was displayed graphically). The rotation of the ring due to unbalanced forces applied by the participants was prevented to make the task easier. Moreover only the 'simultaneous' haptic interactions were supported such that the ring did not move if both subjects did not apply sufficient forces to the ring at the same time.

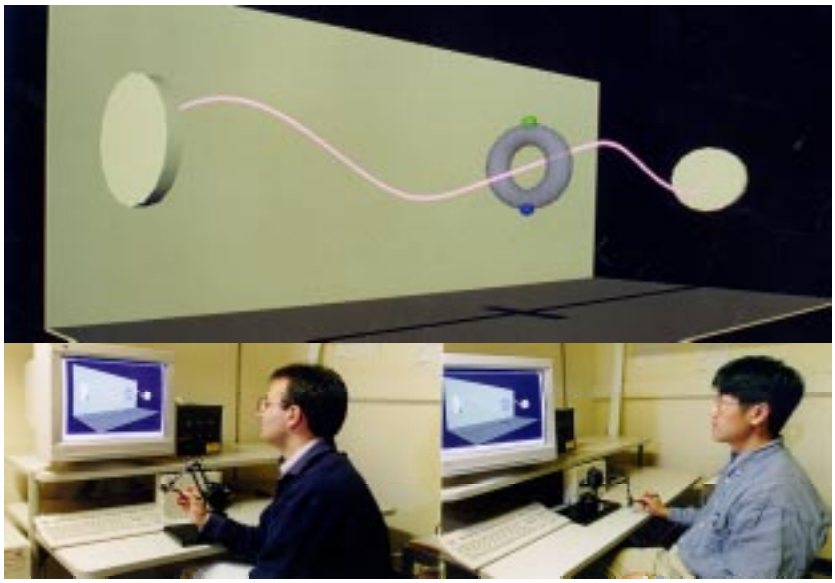


Figure 10. We developed a shared VR set-up that enables two people, at remote locations, to interact with each other through visual and haptic displays. We also designed an experiment to investigate the role of haptics in collaborative manipulation. In our experiment, subjects were asked to hold and 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.

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 virtual environments 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 the user, together with the needs of the task to be accomplished. The 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 the net forces and torques of interaction and give the user the feeling of touching and manipulating objects through a stick or a rigid thimble. Point-based rendering that 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 that computes line-object interaction forces and torques. More recently, limited success has been achieved in simulating 3D object-object interactions where 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 virtual environments are beginning to emerge.

In the future, the 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 the 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. Similar to graphics accelerator cards used today, it is quite likely that much of the 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 does the simulation needs to be to achieve a desired goal.

References

1. Adachi, Y., Kumano, T., Ogino, K. 1995, March 11-15, Intermediate Representation for Stiff Virtual Objects. Proc. IEEE Virtual Reality Annual Intl. Symposium, Research Triangle Park, N. Carolina, pp. 203-210.
2. Adams, R., Hannaford, B., 1998, A Two-Port Framework for the Design of Unconditionally Stable Haptic Interfaces, IEEE/RSJ International Conference on Intelligent Robots and Systems, Victoria, B.C., Canada.
3. Avila, R. S. and Sobierajski, L. M., 1996, A Haptic Interaction Method for Volume Visualization, IEEE Proceedings of Visualization, pp. 197-204.
4. Balniuk, R. and Costa, I. F., 2000, LEM – An Approach for Physically Based Soft Tissue Simulation Suitable for Haptic Interaction, Proceedings of the Fifth Phantom Users Group Workshop.
5. Baraff, D., 1997, "An Introduction to Physically Based Modeling: Rigid Body Simulation I- Unconstrained Rigid Body Dynamics", SIGGRAPH'97 Tutorial Notes.
6. Baraff, D. (1994) Fast Contact Force Computation for Nonpenetrating Rigid Bodies. *ACM (Proceedings of SIGGRAPH)*, 28, 23-34.
7. Basdogan, C., Ho, C., and 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*, Dallas, TX, 61, 77-84.
8. Basdogan C., Ho, C., Srinivasan, M.A., Small, S., Dawson, S., 1998, "Force interactions in laparoscopic simulations: haptic rendering of soft tissues" *Medicine Meets Virtual Reality (MMVR'6) Conference*, pp. 385-391, San Diego, CA, January 19-22.
9. Basdogan C., 1999, " Force Reflecting Deformable Objects for Virtual Environments" in Haptics: From Basic Principles to Advanced Applications, SIGGRAPH' 99, 26 th International Conference on Computer Graphics and Interactive Techniques, Course No: 38, August 8-13, Los Angeles.
10. 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, http://www.amainc.com/MMVR/MMVR2000pro.html
11. Basdogan, C., Ho, C., Srinivasan, M.A., 2001, "Virtual Environments for Medical Training: Graphical and Haptic Simulation of Common Bile Duct Exploration", submitted to the IEEE/ASME Transactions on Mechatronics.
12. 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*, San Diego, CA, January 19-22, 385-391.
13. Basdogan, C., 2001, Real-time Simulation of Dynamically Deformable Finite Element Models Using Modal Analysis and Spectral Lanczos Decomposition Methods, Medicine Meets Virtual Reality (in press).
14. Basdogan, C., Ho, C., Srinivasan, M.A., Slater, M., 2000, "An Experimental Study on the Role of Touch in Shared Virtual Environments" to appear in ACM Human Computer Interactions ".
15. Bathe, K., (1996), "Finite Element Procedures", Prentice Hall, New Jersey.
16. Bier, E.A., Sloan K.R. (1986). Two-Part Texture Mapping. *IEEE Computer Graphics and Applications*, 40-53.
17. Blinn, J.F. (1978). Simulation of Wrinkled Surfaces. *ACM (Proceedings of SIGGRAPH)*, 12(3), 286-292.

18. Buttolo, P., Oboe, R., Hannaford, B., 1997 'Architectures for Shared Haptic Virtual Environments,' Computers and Graphics, vol. 21, pp. 421-9, July-Aug 1997.
19. 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.
20. Colgate, J.E., Brown, J.M., 1994, Factors affecting the z-width of a haptic display", Proceedings of IEEE Int. Conference on Robotics and Automation, 3205-3210.
21. Cotin, S., Delingette, H., Ayache, N., 1999, "Real-time Elastic Deformations of Soft Tissues for Surgery Simulation", IEEE Transactions on Visualization and Computer Graphics, Vol. 5, No. 1, pp. 62-73.
22. 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, pp. 65-78.
23. Dachille, F., Qin, H., Kaufman, A., El-sana J., 1999, "Haptic sculpting of dynamic surfaces", ACM Symposium on Interactive 3D Graphics.
24. Dang, T., Annaswamy, T. M., and Srinivasan, M. A., 2001, Development and Evaluation of an Epidural Injection Simulator with Force Feedback for Medical Training, Medicine Meets Virtual Reality (in press).
25. De, S. and Srinivasan, M. A., 1999, Thin Walled Models for Haptic and Graphical Rendering of Soft Tissues in Surgical Simulations, Medicine Meets Virtual Reality (Eds: Westwood, et al.), IOS Press.
26. De, S. Kim, J. and Srinivasan, M. A., 2001, A Meshless Numerical Technique for Physically Based Real Time Medical Simulations, Medicine Meets Virtual Reality (in press).
27. Ellis, R.E., Sarkar, N. and 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.
28. Ebert, D.S., Musgrave F.K., Peachey D., Perlin K., Worley S. (1994). *Texturing and Modeling*. AP Professional, Cambridge, MA.
29. Edwards, J., Luecke, G., 1996, "Physically based models for use in a force feedback virtual environment", Japan/USA Symposium on Flexible Automation, ASME, pp. 221-228.
30. Foley, J.D., van Dam, A., Feiner, S. K., Hughes, J.F. (1995). *Computer Graphics: Principles and Practice*. Addison-Wesley.
31. Fritz and Barner (1996). Haptic Scientific Visualization. *Proceedings of the First PHANToM Users Group Workshop*, Eds: Salisbury J.K. and Srinivasan M.A. MIT-AI TR-1596 and RLE TR-612.
32. Fukui, Y. (1996) Bump mapping for a force display, Proc. of the First PHANToM User Group Workshop (PUG 96), Dedham, http://www.ai.mit.edu/conferences/pug/pug-proceedings.html
33. Gottschalk, S., Lin, M., and Manocha, D. (1996). OBB-Tree: A hierarchical Structure for Rapid Interference Detection. *ACM (Proceedings of SIGGRAPH),* August.
34. Green, D. F. and Salisbury, J. K. (1997). Texture Sensing and Simulation Using the PHANToM: Towards Remote Sensing of Soil Properties. *Proceedings of the Second PHANToM Users Group Workshop*, Oct. 19-22.
35. 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, "International Journal of Robotics" (*special issue on Tactile Perception*), Vol. 19, No. 7, pp. 668-684.
36. 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", October'99 Vol. 8, No. 5, pp. 477-491, *Presence: Teleoperators and Virtual Environments.*

37. Hollerbach J. and Johnson, D., 2001, to appear in “Human and Machine Haptics” (Eds: Srinivasan, Cutkosky, Howe, and Salisbury).
38. Hubbard, P. (1995). Collision Detection for Interactive Graphics Applications. *IEEE Transactions on Visualization and Computer Graphics*, 1(3), 219-230.
39. Hsu W.M. Hughes J.F., Kaufman H., 1992, “Direct Manipulation of Free-Form Deformations”, *Computer Graphics (Proceedings of the SIGGRAPH)*, Vol. 26, No.2, pp. 177-184.
40. James, D., Pai, D., 1999, ARTDEFO: Accurate Real Time Deformable Objects, In Computer Graphics (SIGGRAPH 99 Conference Proceedings).
41. Krebs, H. I., Hogan, N., Aisen, M. L.,and Volpe, B. T. (1998) Robot-Aided Neurorehabilitation. IEEE Transactions on Rehabilitation Engineering, 6(1): 75-86.
42. Latimar, C., 1997, “Haptic Interaction with Rigid Objects Using Real-time Dynamic Simulation”, M.S. Thesis, Massachusetts Institute of Technology, Mechanical Engineering.
43. Lee, Y., Terzopoulos, D., Waters, K., 1995, “Realistic Modeling for Facial Animation”, *(Proceedings of the SIGGRAPH)*, pp. 55-62.
44. Lin, M. (1993). Efficient Collision Detection for Animation and Robotics. Ph.D. thesis, University of California, Berkeley.
45. 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, Vol. 58 of DSC, pages 459-467.
46. Mandelbrot, B. (1982). *The Fractal Geometry of Nature*. W.H. Freeman.
47. 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, August, 447-452.
48. Max, N.L., Becker, B.G. (1994). Bump Shading for Volume Textures. *IEEE Computer Graphics and App*., 4, 18-20.
49. McNeely, W.A., Puterbaugh K.D., Troy, J.J., 1999, Six Degree-of-Freedom Haptic Rendering Using Voxel Sampling, Proceedings of SIGGRAPH, pp. 401-408.
50. Minsky, M., Ming, O., Steele, F., Brook, F.P., and Behensky, M. (1990). Feeling and seeing: issues in force display. *Proceedings of the symposium on 3D Real-Time Interactive Graphics*, 24, 235-243.
51. Mirtich, B. (1996). Impulse-based Dynamic Simulation of Rigid Body Systems. Ph.D. thesis, University of California, Berkeley.
52. Mirtich, B., Canny, J. (1995). Impulse-based Simulation of Rigid Bodies. Proceedings of Symposium on Interactive 3D Graphics, April.
53. 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.
54. Ng, H., Grimsdale, R., 1996, “Computer Graphics Techniques for Modeling Cloth”, *IEEE Computer Graphic and Applications*, September, pp. 28-41.
55. Perlin, K. (1985). An Image Synthesizer. *ACM SIGGRAPH*, 19(3), 287-296.
56. Ruspini, D.C., Kolarov, K., Khatib O. (1996). Robust Haptic Display of Graphical Environments. Proceedings of the First PHANToM Users Group Workshop, Eds: Salisbury J.K. and Srinivasan M.A. MIT-AI TR-1596 and RLE TR-612.
57. Ruspini, D.C., Kolarov, K., Khatib O. (1997). The Haptic Display of Complex Graphical Environments. *ACM (Proceedings of SIGGRAPH)*, July, 345-352.
58. Salcudean, S. E. and Vlaar, T. D. (1994). On the Emulation of Stiff Walls and Static Friction with a Magnetically Levitated Input/Output Device. *ASME* DSC, 55(1), 303 - 309.

59. Salisbury, J.K., Brock, D., Massie, T., Swarup, N., Zilles C. (1995). Haptic Rendering: Programming touch interaction with virtual objects. *Proceedings of the ACM Symposium on Interactive 3D Graphics*, Monterey, California.
60. Salisbury, J. K., Srinivasan, M. A. (1997). Phantom-Based Haptic Interaction with Virtual Objects. IEEE Computer Graphics and Applications, 17(5).
61. Salisbury, J.K., and Tarr, C., 1997, Haptic Rendering of Surfaces Defined by Implicit Functions, *Proceedings of the ASME*, DSC-61, pp. 61-67.
62. Sederberg, T.W., Parry S.R., 1986, "Free-form Deformation of Solid Geometric Models", *SIGGRAPH Proceedings on Computer Graphics*, Vol. 20, No. 4, pp. 151-160.
63. Siira, J., Pai D. K. (1996). Haptic Texturing - A Stochastic Approach. *Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis, Minnesota, 557-562.
64. Srinivasan MA, Whitehouse JM and LaMotte RH, 1990, Tactile detection of slip: Surface microgeometry and peripheral neural codes, J. Neurophysiology, Vol. 63, No. 6, pp.1323-1332.
65. Srinivasan MA, Beauregard GL, and Brock, DL, 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.
66. Srinivasan, M.A., and Basdogan, C. (1997). Haptics in Virtual Environments: Taxonomy, Research Status, and Challenges. Computers and Graphics, 21(4), 393 – 404.
67. 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.
68. Swarup, N., 1995, "Haptic Interaction with Deformable Objects Using Real-Time Dynamic Simulation", M.S. Thesis, Mechanical Engineering Department, Massachusetts Institute of Technology.
69. Thompson, T.V., Johnson, D.E. and Cohen, E., 1997, Direct haptic rendering of sculptured models, Proc. Sym. Interactive 3D Graphics, Providence, RI, pp. 1-10.
70. Turk, G. (1991). Generating Textures on Arbitrary Surfaces Using Reaction-Diffusion. *ACM (Proceedings of SIGGRAPH)*, 25(4), 289-298.
71. Watt, A., Watt, M. (1992). *Advanced Animation and Rendering Techniques*. Addison-Wesley, NY.
72. Wilson J., Kline-Schoder, Kenton, M.A., Hogan, N. 1999, Algorithms for Network-Based Force Feedback, Proceedings of the Fourth PHANTOM Users Group Workshop, Salisbury, J.K. and Srinivasan, M.A. (Eds), AI Lab Technical Report No. 1675 and RLE Technical Report No. 633, MIT.
73. Wijk, J. J. V. (1991). Spot Noise. *ACM (Proceedings of SIGGRAPH)*, 25(4), 309-318.
74. Witkin, A., and Kass, M. (1991). Reaction-Diffusion Textures. *ACM (Proceedings of SIGGRAPH)*, 25(4), 299-308.
75. Witkin A., Barraff D., Kass M., Tutorial notes on " *An Introduction to Physically-Based Modeling*", SIGGRAPH'98.
76. Worley, S. (1996). A Cellular Texture Basis Function. *ACM (Proceedings of SIGGRAPH)*, August, 291-294.
77. Zilles, C.B., and 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*, IROS, 3, 146-151.