3-DOF HAPTIC RENDERING

C. Basdogan[*], S.D. Laycock[+] A.M. Day[+], V. Patoglu[**], R. B. Gillespie[++]

[*] *Koc University, Istanbul, 34450, Turkey*
[+] *University of East Anglia, Norwich, Norfolk, NR4 7TJ, UK*
[**] *Sabanci University, Istanbul, 34956, Turkey*
[++] *University of Michigan, Ann Arbor, MI, 48109, USA*

1. Introduction

Haptic rendering enables a user to touch, feel, and manipulate virtual objects through a haptic interface (see Figure 1). The field has shown a significant expansion during the last decade. Since the publication of earlier review papers (Salisbury and Srinivasan, 1997; Srinivasan and Basdogan, 1997), new rendering techniques and several new applications have emerged (see the more recent reviews in Basdogan and Srinivasan, 2002, Salisbury et al, 2004, and Laycock and Day, 2007). The applications now cover a wide range of fields including medicine (surgical simulation, tele-medicine, haptic user interfaces for blind persons, rehabilitation for patients with neurological disorders, dental medicine) art and entertainment (3D painting, character animation, digital sculpting, virtual museums), computer-aided product design (free-form modeling, assembly and disassembly including insertion and removal of parts), scientific visualization (geophysical data analysis, molecular simulation, flow visualization), and robotics (path planning, micro/nano tele-manipulation). This chapter will primarily focus on the fundamental concepts of haptic rendering with some discussion of implementation details. In particular, we will focus on the haptic interactions of a single point (i.e. Haptic Interface Point-HIP- in Figure 1) with a 3D object in virtual environments. This corresponds to feeling and exploring the same object through a stylus in real world. This chapter is entitled 3-dof haptic rendering to differentiate it from 6-dof rendering which involves object-object interaction and is covered in the other parts of this book.

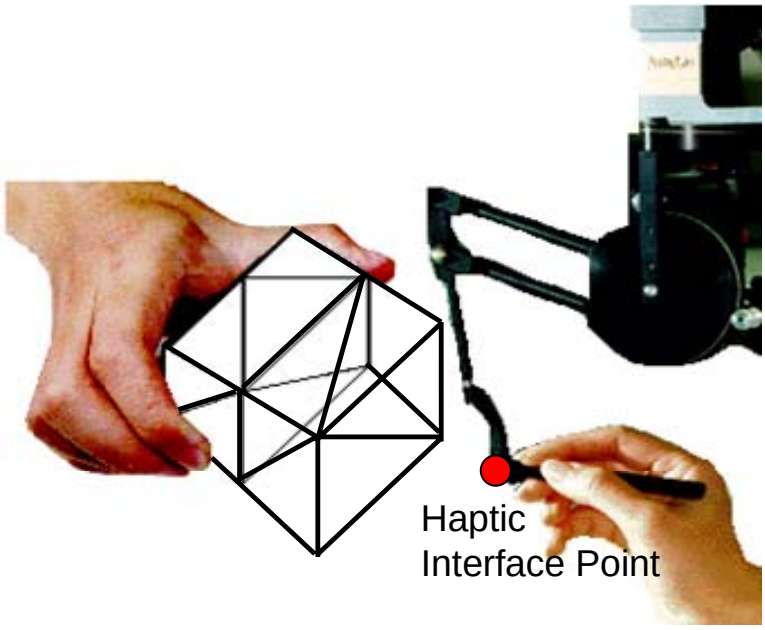


Figure 1. Point-based haptic interactions with 3D objects in virtual environments.

2. Human-Machine Coupling

Haptic rendering is significantly more complex (and interesting) than visual rendering since it is a *bilateral process*: display (rendering) cannot be divorced from manipulation. Thus any haptic rendering algorithm is intimately concerned with tracking the inputs of the user as well as displaying the haptic response. Also note that haptic rendering is computationally demanding due to the high sampling rates required. While the visual system perceives seamless motion when flipping through 30 images per second, the haptic system requires signals that are refreshed at least once every millisecond. These requirements are driven by human haptic ability to detect vibrations which peaks at about 300 Hz but ranges all the way up to 1000 Hz. Note that vibrations up to 1000 Hz might be required to simulate fast motion over fine texture, but also might be required for sharp, impulsive rendering of a changing contact condition.

Haptic rendering requires a haptic interface, a computationally mediated virtual environment, and a control law according to which the two are linked. Figure 2 presents a schematic view of a haptic interface and the manner in which it is most commonly linked to a virtual environment. On the left portion of the figure, mechanical interaction takes place between a human and the haptic interface device, or more specifically, between a fingertip and the device end-effector. In the computational domain depicted on the right, an image of the device end-effector *E* is connected to a proxy *P* through what is called the *virtual coupler*. The proxy *P* in turn interacts with objects such as *A* and *B* in the virtual environment. Proxy *P* might take on the shape of the fingertip or a tool in the user's grasp.

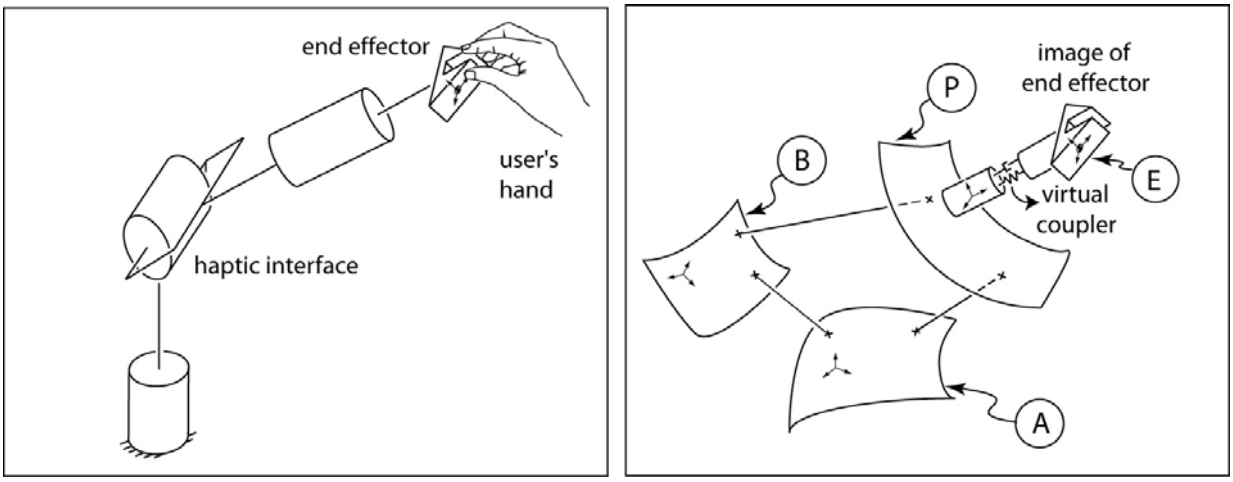


Figure 2. This two part figure presents a schematic representation of haptic rendering. The left figure corresponds to the physical world where a human interacts with the haptic device. The figure on the right depicts the computationally implemented virtual environment.

The virtual coupler is depicted as a spring and damper in parallel, which is a model of its most common computational implementation, though generalizations to 3D involve additional linear and rotary spring-damper pairs not shown. The purpose of the virtual coupler is two-fold. First, it links a forward-dynamics model of the virtual environment with a haptic interface designed for impedance-display[1]. Relative motion (displacement and velocity) of the

[1] Impedance display describes a haptic interface that senses motion and sources forces and moments. An admittance display sources motion and senses forces and moments. Most haptic interface devices are controlled using impedance display, which may be implemented using low inertia motors and encoders connected to the mechanism through low friction, zero backlash, direct-drive or near-unity mechanical advantage transmissions. Impedance display does not require force or torque sensors.

two ends of the virtual coupler determines, through the applicable spring and damper constants, the forces and moments to be applied to the forward dynamics model and the equal and opposite forces and moments to be displayed by the haptic interface. Note that the motion of *P* is determined by the forward dynamics solution, while the motion of *E* is specified by sensors on the haptic interface. The second role of the virtual coupler is to filter the dynamics of the virtual environment so as to guarantee stability when display takes place through a particular haptic device. The parameters of the virtual coupler can be set to guarantee stability when parameters of the haptic device hardware are known and certain input-output properties of the virtual environment are met. Thus the virtual coupler is most appropriately considered part of the haptic interface rather than part of the virtual environment (Adams and Hannaford, 1999, 2002). If an admittance-display architecture is used, an alternate interpretation of the virtual coupler exists, though it plays the same two basic roles. For further discussion of the virtual coupler in performance/stability tradeoffs in either the impedance or admittance-display cases, see the work done by Adams and Hannaford (1999, 2002), and Miller and Colgate (2000).

One final note can be made with reference to Figure 2: rigid bodies in the virtual environment, including *P*, have both configuration and shape -they interact with one another according to their dynamic and geometric models. Configuration (including orientation and position) is indicated in Figure 2 using reference frames (three mutually orthogonal unit vectors) and reference points fixed in each rigid body. Shape is indicated by a surface patch. Note that the image of the device end-effector *E* has configuration but no shape. Its interaction with *P* takes place through the virtual coupler and requires only the configuration of *E* and *P*.

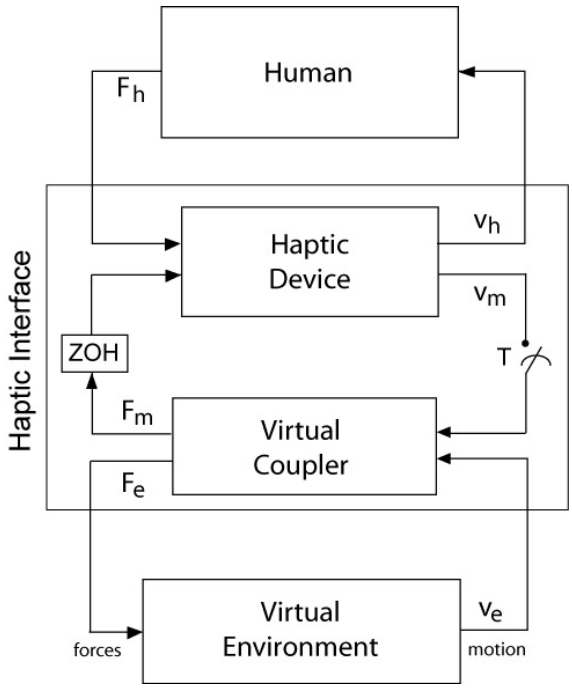


Figure 3. A block diagram of haptic rendering according to an impedance display architecture. There are three major blocks in the diagram modeling input/output characteristics of the human, the haptic interface and the virtual environment.

The various components in Figure 2, including the human user, haptic device, virtual coupler, and virtual environment, form a coupled dynamical system whose behavior depends on the force/motion relationship established by the interconnection of the components. Figure 3

shows these components interconnected in a block diagram, where the additional indication of causality has been made. Causality expresses which force and motion variables are inputs and which are outputs for each component. For example, the human operates on the velocity v_h (common to the finger and end-effector) to produce the force F_h imposed on the haptic device. The haptic device is a two-port that operates on the force F_h imposed by the human and the force F_m produced by its motors to produce the velocities v_h and v_m. Usually, by careful haptic device transmission design, v_h and v_m are the same, and measured with a single encoder. Intervening between the human and haptic device, that 'live' in the continuous, physical world and the virtual coupler and virtual environment that live in the discrete, computed world are a sampling operator T and zero-order hold (ZOH). The virtual coupler is shown as a two-port that operates on velocities v_m and v_e to produce the motor command force F_m and force F_e imposed on the virtual environment. Forces F_m and F_e are usually equal and opposite. Finally, the virtual environment is shown in its forward dynamics form, operating on applied forces F_e to produce response motion v_e. Naturally, the haptic device may use motors on its joints, so the task-space command forces F_m must first be mapped through the manipulator Jacobian before being applied to the motors.

Note that the causality assumption for the human is by itself rather arbitrary. However, causality for the haptic device is essentially determined by electro-mechanical design, and causality for the virtual coupler and virtual environment is established by the implementation of a discrete algorithm. The causality assumptions in Figure 3 correspond to impedance display. Impedance display is the most common but certainly not the only possible implementation. See (Adams and Hannaford, 1999) for a framework and analysis using network diagrams (that do not indicate causality), which is more general.

3. Rendering of 3D Rigid Objects

Typically, a haptic rendering algorithm is made of two parts: (a) collision detection and (b) collision response (see Figure 4). As the user manipulates the probe of the haptic device, the new position and orientation of the haptic probe are acquired and 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

…
```

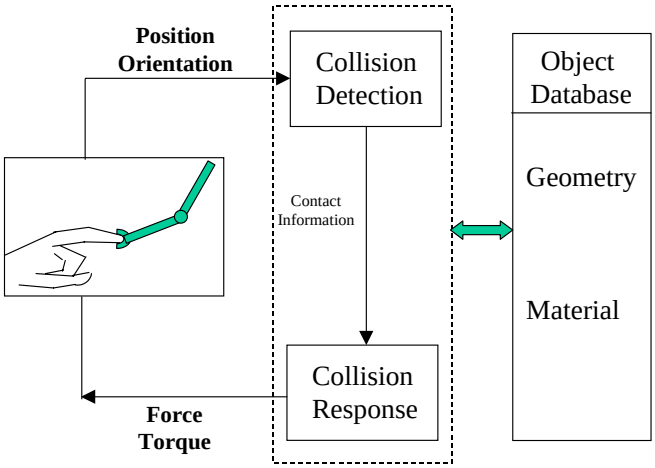
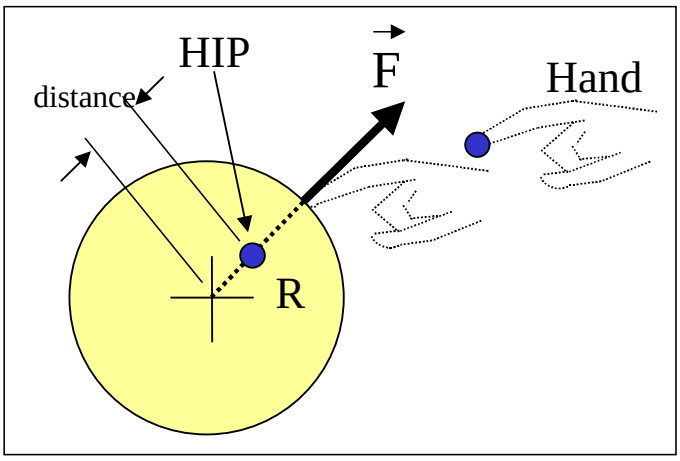


Figure 4. A haptic interaction algorithm is typically made of two parts: (a) collision detection and (b) collision response. The haptic loop seen in the figure requires an update rate of around 1 kHz for stable force interactions. Computationally fast collision detection and response techniques are necessary 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 5). 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 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 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 5.




```
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 5. 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 is 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:

$$\vec{F} = k\,\Delta\vec{x} \qquad (1)$$

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 feel soft, setting a high value can make the interactions unstable by causing undesirable vibrations. Figure 6 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, Ellis et al., 1996, and Gillespie and Cutkosky 1996).

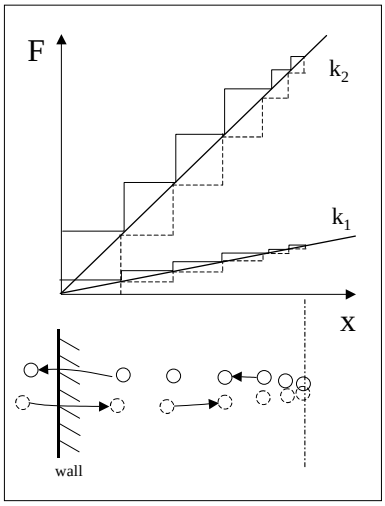


Figure 6. Force-displacement curves for touch interactions with real and virtual walls. In the case of real wall, the force-displacement curve is continuous. However, we see the "staircase" effect when simulating touch interactions with a virtual wall. This is due to the fact that a haptic device can 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_1 and k_2). On the other hand, a low value of stiffness coefficient generates a soft wall, which is not desirable either.

Although the basic recipe for haptic rendering of virtual objects seems easy to follow, rendering 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 the real world. However, several haptic rendering techniques have been developed to simulate complex touch interactions in virtual environments. 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; Nelson et al., 1999; Gregory et al. 2001; Johnson et al. 2003; Laycock et al. 2005; Otuday and Lin, 2005). 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. 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), god-object, proxy point, or surface contact point. 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 similar to that experienced when exploring the objects in real environments with the tip of a stylus.

An important component of any haptic rendering algorithm is the collision response. 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 in haptic rendering to accurately compute the interaction forces that will be reflected to the user through the haptic device (Ho et al., 1999; Basdogan and Srinivasan, 2002). In other words, the computation of IHIP relies on the contact history. 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. For example, Figure 7a shows a thin object with the HIP positioned at three successive time steps. Step 1 shows the HIP on the left hand side just coming into contact with the thin object. At Step 2, the HIP has penetrated the thin object and is now closer to the right face of the object. The HIP will be forced out the other side producing an undesired result. A similar problem will occur if the HIP is located equidistant to two faces of the virtual object. Figure 7b illustrates this case and it is unclear which face normal to choose by looking at a single time step. The HIP could easily be forced out of the object in the incorrect direction. To overcome these problems an approach is required to keep a contact history of the position of the HIP. The next section discusses techniques that, among other advantages, overcome these problems.

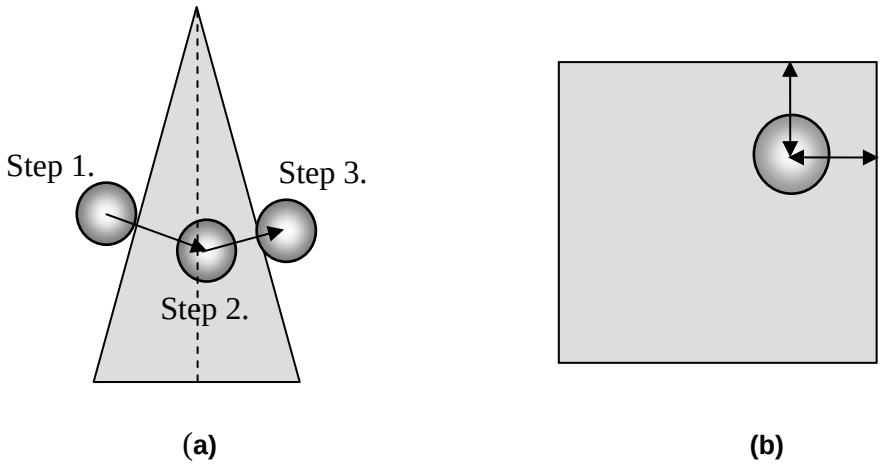


Figure 7. a) The Haptic Interface Point can be forced out of the wrong side of thin objects. **b)** The Haptic Interface Point is equidistant to both faces. The algorithm is unable to decide which face is intersected first by looking at a single time step.

The algorithms developed for 3-dof point-based haptic interaction depend on the geometric model of the object: 1) surface models and 2) volumetric models. The surface models can be also grouped as 1) polygonal surfaces, 2) parametric surfaces, and 3) implicit surfaces.

3.1. Surface Models

Polygonal surfaces: Virtual objects have been modeled using polygonal models in the field of computer graphics for decades due to their simple construction and efficient graphical rendering. For similar reasons haptic rendering algorithms were developed for polygonal models and triangular meshes in particular. Motivating this work was the ability to directly augment the existing visual cues with haptic feedback utilizing the same representations.

The first method to solve the problems of single point haptic rendering for polygonal models was developed at the Massachusetts Institute of Technology (MIT) (Zilles and Salisbury, 1995). The method enabled a contact history to be kept and at the time it was able to provide stable force feedback interactions with polygonal models of 616 triangular faces on a computer with a 66MHz Pentium processor. A second point known as the "god-object" was employed to keep the contact history. It would always be collocated with the HIP if the haptic device and the virtual object were infinitely stiff. In practice the god-object and the HIP are collocated when the HIP is moving in free space. As the HIP penetrates the virtual objects the god-object is constrained to the surface of the virtual object. An approach is subsequently required to keep the god-object on the surface of the virtual object as the HIP moves around inside. Constraint based approaches that keep a point on the surface sometimes refer to this point as the Surface Contact Point, SCP. The position of the god-object can be determined by minimizing the energy of a spring between the god-object and the HIP, taking into account constraints represented by the faces of the virtual object. By minimizing L in Equation (2) the new position can be obtained. The first line of the equation represents the energy in the spring and the remaining three lines represent the equations of three constraining planes. The values x, y and z are the coordinates of the god-object and x_p, y_p and z_p represent the coordinates of the HIP.

$$
\begin{aligned}
L = {} & \frac{1}{2}(x - x_p)^2 + \frac{1}{2}(y - y_p)^2 + \frac{1}{2}(z - z_p)^2 \\
& + l_1(A_1 x + B_1 y + C_1 z - D_1) \\
& + l_2(A_2 x + B_2 y + C_2 z - D_2) \\
& + l_3(A_3 x + B_3 y + C_3 z - D_3)
\end{aligned}
\qquad (2)
$$

where, L = value to be minimized.
l_1, l_2, l_3 = Lagrange multipliers
A, B, C, D = coefficients for the constraint plane equations.

To efficiently solve this problem a matrix can be constructed and used in Equation (3) to obtain the new position of the god-object as given by x, y and z. When there are three constraining planes limiting the motion of the god-object the method only requires 65 multiplications to obtain the new coordinates. The problem is reduced as the number of constraint planes is reduced.

$$\begin{pmatrix} 1 & 0 & 0 & A_1 & A_2 & A_3 \\ 0 & 1 & 0 & B_1 & B_2 & B_3 \\ 0 & 0 & 1 & C_1 & C_2 & C_3 \\ A_1 & B_1 & C_1 & 0 & 0 & 0 \\ A_2 & B_2 & C_2 & 0 & 0 & 0 \\ A_3 & B_3 & C_3 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ l_1 \\ l_2 \\ l_3 \end{pmatrix} = \begin{pmatrix} x_p \\ y_p \\ z_p \\ D_1 \\ D_2 \\ D_3 \end{pmatrix} \quad (3)$$

It was noted by Ruspini et al. (1997) that small numerical inconsistencies can cause gaps in polygonal models that enable the god-object to slip into the virtual objects. To overcome this the topology of the surface must be reconstructed. To avoid this reconstruction phase a strategy similar in style to the previous approach was developed which permitted the constrained point to have finite size (Ruspini et al., 1997). The algorithms employed by Ruspini et al. were originally developed for the robotics field. The term Virtual Proxy is used to refer to the spherical object constrained to the surface of the virtual objects. Its motion is akin to the motion of a robot greedily attempting to move towards a goal, in this case the HIP. Configuration space obstacles are constructed by wrapping the virtual object by a zone equal to the radius of the Virtual Proxy. Doing this enables a single point to be incorporated as the haptic probe once more. Lagrange multipliers can then be used as described by Zilles and Salisbury (1995) to obtain the new position of the Virtual Proxy.

A more procedural constraint based approach was developed by Ho et al. (1999) for single point rendering. They state that it increases the servo-rate, facilitates stable haptic interactions and importantly enables the servo rate to be independent of the number of polygons. They refer to their constrained point as the Ideal Haptic Interface Point, IHIP, with a force being sent to the haptic device based on a spring between the IHIP and HIP. Figure 8 illustrates an overview of the algorithm. A two-dimensional slice of a three-dimensional object has been included to represent the virtual object. Initially the IHIP and the HIP are set at the same position identical to the god-object and Virtual Proxy approaches. At each time step a line segment is constructed from the previous HIP, P_HIP, to the current HIP. If there exists an intersection point between this line segment and the virtual object, then the IHIP is constrained to the closest point to the current HIP on the face nearest to the P_HIP. This is depicted in Figure 8b. As the HIP moves the IHIP is tracked over the surface of the mesh by choosing the closest feature to the HIP. For efficiency, only those features (edge, vertex, face) that bound the current feature are tested, as is depicted in Figures 8c and 8d. When the vector from the IHIP to HIP points in the direction of the current feature normal, then there is no contact with the surface and the HIP and IHIP are once again collocated.

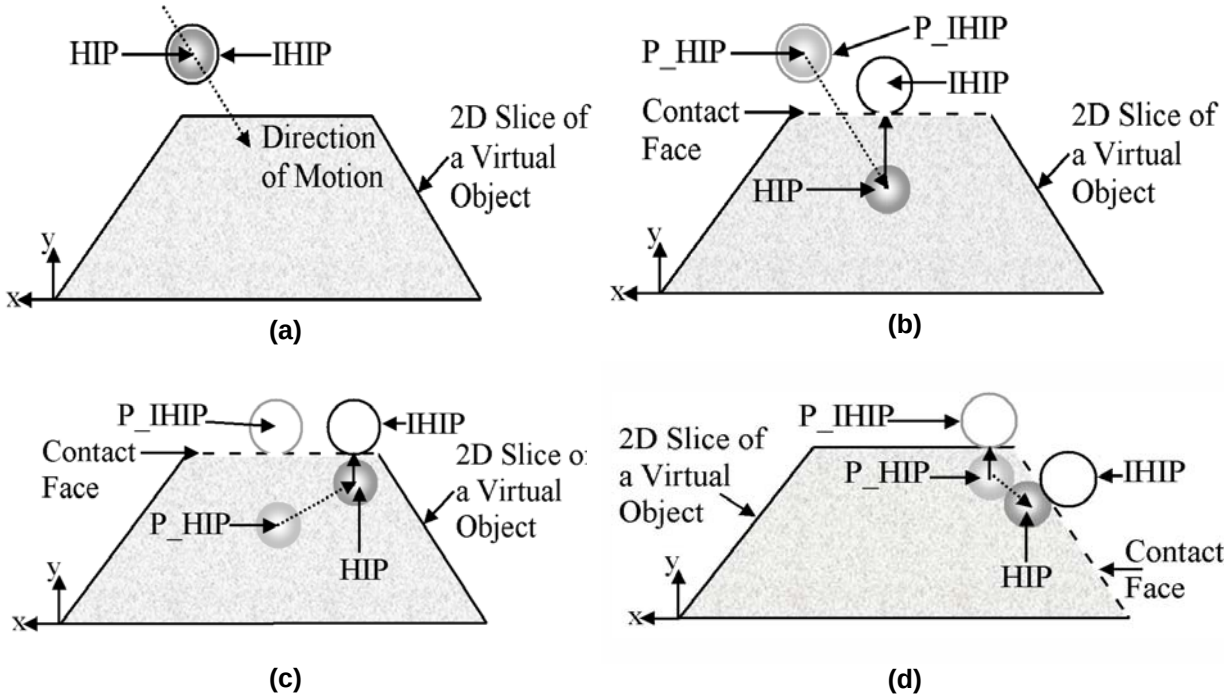


Figure 8 a) The HIP and IHIP illustrated moving down towards the top of the two-dimensional slice of the virtual object. **b)** The line segment between the P_HIP and the HIP intersects with the virtual objects. The contact face is shown with the dotted line and the IHIP is constrained to the surface. **c)** The IHIP is tracked along the surface of the virtual object. **d)** The HIP is now closer to a new feature of the virtual object and so the contact face is updated.

Parametric surfaces: Polygonal representations are perfect for displaying simple objects, particularly those with sharp corners, but are limited when it comes to representing highly curved objects. In such case a large number of polygons would be required to approximate the curved surface, resulting in higher memory requirements. Modeling interactively using these models can also be very tedious. To overcome this, parametric surfaces have been used and are very important in modeling packages and Computer Aided Design (CAD) packages.

To directly render parametric models without performing a difficult conversion into a polygonal representation, haptic rendering algorithms have been developed to allow direct interaction with NURBS surfaces. In 1997, at the University of Utah, Thompson et al. (1997) developed a technique for the haptic rendering of NURBS surfaces. The motivation was to be able to interact with a CAD modeling system using the Sarcos force-reflecting exoskeleton arm. The algorithm is broken into two phases. Firstly, the collision detection between the HIP and the surfaces is undertaken and secondly they employ a Direct Parametric Tracing algorithm to constrain a point to the surface as the HIP is permitted to penetrate the surface. The constrained point will be referred to as the Surface Contact Point, SCP. The first stage of the collision detection uses bounding boxes encompassing the surfaces to aid in trivial rejection. If the HIP is inside the bounding box then the HIP is projected onto the control mesh. The parameters (u, v) are defined for each vertex of the control mesh. The (u, v) parameters for the projected point can be obtained by interpolating the parameter values at the vertices. The distance between the HIP and the point on the surface can then be obtained. The Direct Parametric Tracing method tracks the position of the HIP on the surface. As the HIP moves it is projected onto the surface tangent plane, tangential to the gradient of the surface at the previous location of the SCP. The new SCP and tangent plane are then found by

parametric projection using the projected HIP (Thompson et al., 1997). The force returned to the device is based on a spring damper model between the HIP and the surface.

Alternatively, the minimum distance between convex parametric surfaces may be determined by formulating a nonlinear control problem and solving it with the design of a switching feedback controller (Patoglu and Gillespie, 2004, 2005). The controller simply has the job of stabilizing the integration of the differential kinematics of the error vector that connects two candidate points, one drawn from each of two interacting surfaces. The controller manipulates the parameters (u, v) and (r, s) of the candidate points until the projections of the error vector onto all four surface tangents are driven to zero. With the design of a suitable feedback control law, the simulation of the differential kinematics produces an asymptotically convergent algorithm. While algorithms based on Newton's Iteration have a limited region of attraction, the algorithm built around the control formulation guarantees global uniform asymptotic stability, hence dictates that any pair of initial points belonging to the convex surface patches will converge to the closest point solution without ever leaving the patches (Patoglu and Gillespie, 2005). Global convergence for a narrow phase algorithm greatly simplifies the design of a multi-phase algorithm with global convergence. The algorithm may be run as the surface patches move, where it becomes a tracking algorithm. Other notable features include no requirement for matrix inversion, high computational efficiency, and the availability of analytic limits of performance. Together with a top-level switching algorithm based on Voronoi diagrams, this closest point algorithm can treat parametric models formed by tiling together surface patches (Patoglu and Gillespie, 2005).

One of the most promising applications of rendering parametric surfaces is in free-form design. Free-form surfaces are defined by parametric functions and the conventional methods of design using these surfaces require tedious manipulation of control points and careful specification of constraints. However, the integration of haptics into free form design improves the bandwidth of interactions and shortens the design cycle. Dachille et al. (1999) developed a method that permits users to interactively sculpt virtual B-spline objects with force feedback. In this approach, point, normal, and curvature constraints can be specified interactively and modified naturally using forces. Nowadays, commercial packages based on the free-form design concept (FreeForm Concept and FreeForm Modeling Plus from Sensable Tech.) offer alternative solutions to the design of jewelry, sport shoes, animation characters, and many other products. Using these software solutions and a haptic device, the user can carve, sculpt, push, and pull instead of sketching, extruding, revolving, and sweeping as in traditional design.

Implicit Surfaces: An early approach for the haptic rendering of implicit surfaces was developed by Salisbury and Tarr (1997). Their approach used implicit surfaces defined by analytic functions. Later, Kim et al. (2002) developed a technique where an implicit surface is constructed that wraps around a geometric model to be graphically rendered. To ensure the surface can be accurately felt a virtual contact point is incorporated. This point is constrained to the surface as shown in Figure 9.

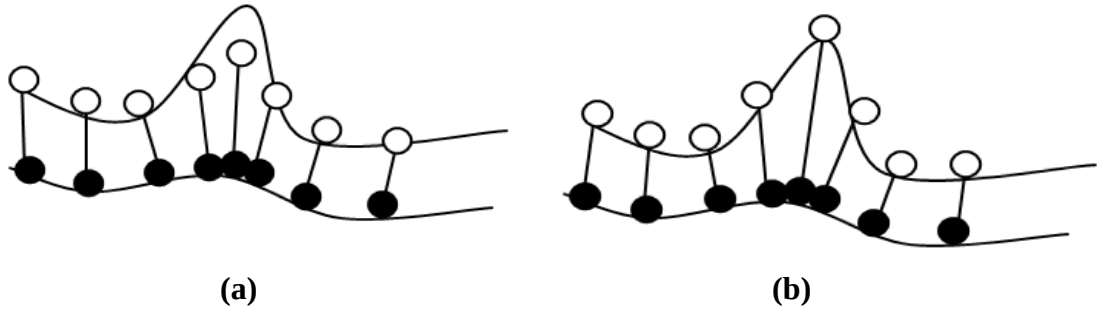


Figure 9. Computing the force magnitude, (a) shows an approximate method and (b) illustrates the approach by Kim et al. Adapted from Kim et al. (2002)

The method was used for virtual sculpting. The space occupied by the object is divided into a three-dimensional grid of boxes, in a similar strategy to the volume rendering techniques that will be discussed in the next section. Collision detection using implicit surfaces can be performed efficiently, since by using implicit surfaces it is possible to determine whether a point is interior, exterior or on the surface by evaluating the implicit function with a point. The potential value is determined for each grid point and then it can be used with an interpolation scheme to return the appropriate force. The surface normal at each grid point is calculated from the gradient of the implicit function. The surface normal for any point can then be determined by interpolating the values of the surface normals at the eight neighbors.

The previous two methods permit only one side of the surface to be touched. For many applications this is not sufficient, since users often require both sides of a virtual object to be touched. Maneewarn et al. (1999) developed a technique using implicit surfaces that enabled the user to interact with the exterior and interior of objects. The user's probe is restricted by the surface when approached from both sides.

3.2. Volumetric Models

Volumetric objects constructed from individual voxels can store significantly more information than a surface representation. The ability to visualize volume data directly is particularly important for medical and scientific applications. There are a variety of techniques for visualizing the volume data such as the one proposed by Lacroute and Levoy for shearing, warping and compositing two-dimensional image slices (Lacroute and Levoy, 1994). In contrast, a method termed “splatting” can be used where a circular object can be rendered in each voxel (Laur and Hanrahan 1991). Each circular object is aligned to the screen and rendered to form the final image. Volume data can also be visualized indirectly by extracting a surface representation using methods such as Marching Cubes. Once the surface is extracted, haptic interaction can then take place using the surface based methods discussed earlier (Eriksson et al. 2005; Körner et al. 1999). However, the process of surface extraction introduces a number of problems. As it requires a preprocessing step the user is prevented from modifying the data during the simulation and it also can generate a large number of polygons. Furthermore, by only considering the surface, it is not possible to incorporate all the structures present in a complex volume. To create a complete haptic examination of volume data, a direct approach is required. Iwata and Noma (1993) were the first to enable haptic feedback in conjunction with volume data using a direct volume rendering approach, which they termed Volume Haptization. The approach illustrated ways of mapping 3D vector or scalar data to forces and torque. The mapping must determine the forces at interactive rates and typically the forces must directly relate to the visualization of the data. This form of direct

volume rendering is particularly useful for scientific visualization (Lawrence et al. 2000). Iwata and Noma used their approach for the haptic interaction of data produced in Computational Fluid Dynamics. In this case force could be mapped to the velocity and torque mapped to vorticity.

Utilizing Computed Tomography (CT) or Magnetic Resonance Imaging (MRI) as a basis for volume rendering enables a three dimensional view of patient specific data to be obtained. Enabling the user to interact with the patient data directly is useful for the medical field particularly since surgeons commonly examine patients through their sense of touch. Gibson (1995) developed a prototype for the haptic exploration of a 3D CT scan of a human hip. The CT data is converted to voxels with each voxel incorporating information for both haptic rendering and graphical rendering. The human hip is then inserted into an occupancy map, detailing where the model is located in the voxel grid. The occupancy map consists of a regularly spaced grid of cells. Each cell either contains a null pointer or an address of one of the voxels representing an object in the environment. The size of the occupancy map is set to encompass the entire virtual environment. The fingertip position controlled by the haptic device is represented by a single voxel. Collision detection between the fingertip and the voxels representing the human hip are determined by simply comparing the fingertip voxel with the occupancy map for the environment.

Avila and Sobierajski (1996) developed a technique for the haptic rendering of volume data, where the surface normals were obtained analytically. The method works by decomposing the object into a three-dimensional grid of voxels. Each voxel contains information such as density, stiffness and viscosity. An interpolation function is used to produce a continuous scalar field for each property. They present one example of interacting with a set of dendrites emanating from a lateral geniculate nucleus cell. The data was obtained by scanning with a confocal microscope. The additional functionality was developed to enable the user to visualize and interact with the internal structure. Bartz and Guvit (2000) used distance fields to enable the direct volume rendering of a segment of an arterial blood vessel derived from rotational angiography. The first distance field is generated by first computing a path from a given starting voxel to a specified target voxel in the blood vessel. This path is computed using Dijkstra's Algorithm. This creates a distance field which details the cost of traveling to the target voxel. A second field is based on the Euclidean distance between each voxel and the surface boundary. A repulsive force can then be rendered based on the distance between the haptic probe and the surface. The effects of the two distance fields are controlled using constant coefficients. A local gradient at a point can then be obtained using trilinear interpolation of the surrounding voxels. The coefficients of the distance fields must be chosen carefully to avoid oscillations.

Several researchers have investigated cutting and deforming volumetric data representing anatomical structures (Agus et al. 2002; Eriksson et al. 2005; Kusumoto et al. 2006; Petersik et al. 2002; Gibson et al. 1997). Gibson et al. (1997) segmented a series of MRI images by hand for the simulation of arthroscopic knee surgery. The deformation of the model was calculated using an approach which permits a volume to stretch and contract in accordance to set distances (Gibson 1997). The physical properties of the material are also useful for sculpting material represented by volume data. Chen and Sun (2002) created a system for sculpting both synthetic volume data and data obtained from CT, MRI and Ultrasound sources. The direct haptic rendering approach utilized an intermediate representation of the volume data (Chen et al. 2000). The intermediate representation approach to haptic rendering

was inspired from its use in rendering geometric models (Mark et al. 1996). The sculpting tools developed by Chen and Sun were treated as volumes allowing each position in the tool volume to effect the object volume data. They simulated a variety of sculpting effects including melting, burning, peeling and painting.

When interacting with the volume data directly, an approach is required to provide stiff and stable contacts in a similar fashion to the rendering achieved with geometric representations. This is not easily accomplished when using the techniques based on mapping volume data directly to forces and torques. One strategy is to use a proxy constrained by the volume data instead of utilizing an intermediate representation as in the previous example (Ikits et al. 2003; Lundin et al. 2002; Palmerius 2007). Lundin et al. (2002) presented an approach aimed at creating natural haptic feedback from density data with solid content (CT scans). To update the movements of the proxy point, the vector between the proxy and the HIP was split into components: one along the gradient vector (f_n) and the other perpendicular to it (f_t). The proxy could then be moved in small increments along f_t. Material properties such as friction, viscosity and surface penetratability could be controlled by varying how the proxy position was updated. Palmerius (2007) developed an efficient volume rendering technique to encompass a constraint based approach with a numerical solver and importantly a fast analytical solver. The proxy position is updated by balancing the virtual coupler force, $\vec{f}$, against the sum of the forces from the constraints, $\vec{F}_i$. The constraints are represented by points, lines and planes. The balancing is achieved by minimizing the residual term, $\vec{\varepsilon}$, in the following equation:

$$\vec{\varepsilon} = -\vec{f}(\vec{x}_{proxy}) + \sum_i \vec{F}_i(\vec{x}_{proxy}) \qquad (4)$$

By modifying the effects of the constraints in the above equation different modes of volume exploration can take place such as surface-like feedback and 3D friction. Linear combinations of the constraint effects can be used to obtain the combined residual term. An analytical solver may then be used to balance the equation and hence find the position of the proxy. The analytical solver is attempted first for situations where the constraints are orthogonal, however, if this fails a numerical solver is utilized. This combination of techniques is available in the open source software titled Volume Haptics Toolkit (VHTK).

4. Surface Details: Smoothing, Friction, and Texture

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 surfaces in nature are covered with some type of *texture* that is sensed and distinguished quite well by our tactile system. Haptic texture is a combination of small-scale variations in surface geometry and its adhesive and frictional characteristics. Oftentimes, displaying the detailed geometry of textures is computationally too expensive. As an alternative, 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 10).

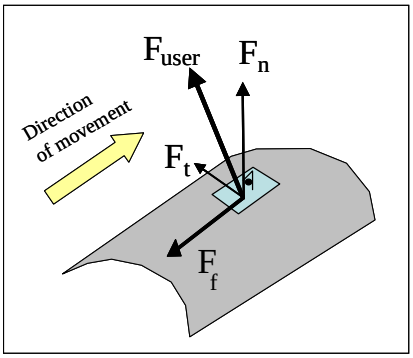


Figure 10. Forces acting on the user ($F_{user} = F_n + F_t + F_f$) during haptic simulation of friction and textures. The normal force can be computed using a simple physics-based model such as Hooke's law ($F_n = k\ \Delta x$, where Δx is the depth of penetration of the haptic probe into the virtual 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 texture, we change the magnitude and direction of the normal vector (F_n) using the gradient of the texture field at the contact point.

Smoothing: A rapid change in surface normals associated with sharp edges between joining surfaces or between faces in a polygonal models causes force discontinuities that may prove problematic during haptic rendering. Incorporating techniques to blend between the surface normals can alleviate these problems. Without this type of technique high numbers of polygons would be required to simulate surfaces with smooth curved areas. Strategies analogous to Gourard or Phong shading, used for interpolating normals for lighting, can be developed for haptic rendering. The paper by Morgenbesser and Srinivasan (1996) was the first to demonstrate the use of force shading for haptic rendering. Using a similar technique to Phong shading Salisbury et al. (1995) found that a smooth model could be perceived from a coarse three-dimensional model. This is akin to visualizing a smooth three-dimensional object using Phong shading when a relatively low number of triangles are used in the underlying geometry.

Ruspini et al. (1997) also incorporated a force shading model, which interpolated the normals similar to Phong shading. A two pass technique was utilized to modify the position of the Virtual Proxy. The first stage computes the closest point, CP, between the HIP and a plane that runs through the previous virtual proxy position. The plane's normal is in the same direction as the interpolated normal. The second stage proceeds by using the CP as the position of the HIP in the usual haptic rendering algorithm described in the previous section. They state that the advantages of this method are that it deals with the issue of force shading multiple intersecting shaded surfaces and that by modifying the position of the Virtual Proxy the solution is more stable.

In some approaches changes in contact information, penetration distance and normals can affect the force feedback significantly between successive steps of the haptic update loop. These changes can cause large force discontinuities producing undesirable force feedback. Gregory et al. (2001) encountered this problem and employed a simple strategy to interpolate between two force normals. Their strategy prevents the difference between previous and current forces becoming larger than a pre-defined value, F_{max}. This simple approach provides a means of stabilizing forces.

$$
\begin{aligned}
&\textit{if } (F_{1x}\text{-}F_{0x}) > 2F_{max} \\
&\qquad \textit{then } F_{1x} = F_{0x} - F_{max} \\
&\textit{else if } (F_{1x} - F_{0x}) > F_{max} \\
&\qquad \textit{then } F_{1x} = (F_{0x} + F_{1x}) / 2 \\
&\textit{return } F_{1x}
\end{aligned}
$$

The pseudo code given above is used to prevent the new force vector, F_1, becoming significantly different from the previous force vector, F_0. The pseudo code presented illustrated how the force smoothing takes place for the *x* components of the force vectors. Similar code is required for the *y* components and *z* components.

Friction: In the previous section the methods for computing forces that act to restore the HIP to the surface of the virtual object have been discussed. If this force is the only one incorporated then the result is a frictionless contact where the sensation perceived is analogous to moving an ice-cube along a glassy surface (Salisbury et al., 1995). Achieving this is a good result as some good properties with respect to the design of the haptic feedback device are exhibited in addition to smooth and stable feedback. However, this interaction is not very realistic, in most cases, and can even hinder the interaction as the user slips off surfaces accidentally. Several approaches have been developed to simulate both static and dynamic friction to alleviate this problem (Salcudean and Vlaar 1994; Salisbury et al., 1995, Mark et al., 1996, Ruspini et al., 1997, Kim et al., 2002). 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.

Salisbury et al. (1995) developed a stick-slip friction model enhancing the feedback from their god-object approach. The model utilizes Coulomb friction and records a stiction point. The stiction point remains static until an offset between the stiction point and the user's position is exceeded. At this stage the stiction point is moved to a new location along a line that connects the previous stiction point and the user's position. Kim et al. (2002) enabled friction to be incorporated with their implicit surface rendering technique. By adjusting the position of the contact point on the surface a component of force tangential to the surface could be integrated. To achieve this, a vector, *V*, is obtained between the previous and new positions of the contact point on the surface. A friction coefficient can be integrated to determine a point, *P*, along *V*. The surface point that is intersected by a ray emanating from the HIP position passing through *P* is chosen as the new contact point.

At the University of North Carolina, Chapel Hill, Mark et al. (1996) developed a model for static and dynamic friction. The surfaces of the objects are populated by snags, which hold the position of the user until they push sufficiently to leave the snag. When the probe moves further than a certain distance from the centre of the snag the probe is released. While stuck in a snag a force tangential to the surface pulls the user to the centre of the snag and when released a friction force proportional to the normal force is applied. It is easily envisaged that this technique is appropriate for representing surface texture by varying the distribution of the snags. Surface texture will be described in the next section.

Texture: 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. There exists a strong correlation between the friction of a surface and its surface roughness, or texture. However, texture enriches the user’s perception of a surface to a higher extent than friction, as extra details about the surface can be perceived. Integrating texture into haptic rendering algorithms presents more information,

to the user, about the virtual object than applying images to the surface of objects for graphical rendering, using texture mapping. Surface texture is important when humans interact with objects and therefore it is important for the haptic rendering of virtual objects. Many researchers have investigated the psychophysics of tactile texture perception. Klatzky et al. (2003) investigate haptic textures perceived through the bare finger and through a rigid probe. Choi and Tan (2004) investigated the perceived instabilities in haptic texture rendering and concluded that the instabilities may come from many sources including the traditional control instability of haptic interfaces as well as inaccurate modeling of environment dynamics, and the difference in sensitivity to force and position changes of the human somatosensory system. Minsky et al. (1990) developed a simulation to represent the roughness of varying degrees of sandpaper. Users were then asked to order the pieces of simulated sandpaper according to their roughness. A texture depth map was created, utilized by the haptic device by pulling the user's hand into low regions and away from high regions. A strategy similar to that of bump mapping objects, utilized in graphical rendering, was employed by Ho et al. (1999). For graphical texture mapping, techniques have been developed to enable statistical approaches to the generation of textures (Siira and Pai, 1996; Fritz and Barner, 1996; Basdogan et al., 1997). Fritz and Barner (1996) developed two methods for rendering stochastic based haptic textures. The lattice texture approach works by constructing a 2D or 3D grid where a force is associated to each point. The second method labeled local space approach also uses a lattice defined in the texture space coordinate system. In this case the forces are determined for the centers of the grid cells. For implicit surfaces Kim et al. (2002) enabled Gaussian noise and texture patterns to directly alter the potential values stored in the points forming three-dimensional grids. The three-dimensional grids encompass the virtual objects. The adaptation could be incorporated without increasing the overall complexity of the haptic rendering algorithm. Fractals are also appropriate for modeling natural textures since many objects seem to exhibit self-similarity Ho et al. (1999) 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.

Recently haptic texturing has also been employed between two polygonal models. This approach can be applied to the haptic rendering techniques for object-object interactions. Otaduy et al. (2004) developed a technique to estimate the penetration depth between two objects described by low resolution geometric representations and haptic textures created from images that encapsulate the surface properties.

5. Summary and Future

The goal of 3-dof haptic rendering is to develop software algorithms that enable a user to touch, feel, and manipulate objects in virtual environments through a haptic interface. 3-dof haptic rendering views the haptic cursor as a point in computing point-object interaction forces. However, this does not restrict us to simulate tool-object or multi-finger interactions. For example, a 3D tool interacting with a 3D object can be modeled as dense cloud of points around the contact region to simulate tool-object interactions. Many of the point-based rendering algorithms have been already incorporated into commercial software products such as the Reachin API[1], GHOST SDK, and OpenHaptics[2]. Using these algorithms, 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

[1] http://www.reachin.se
[2] http://www.sensable.com

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 emerging.

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 (see the review in Basdogan et al., 2004). 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 need to be to achieve a desired goal.

References:

Adachi, Y., Kumano, T., Ogino, K., (1995). “Intermediate representation for stiff virtual objects”, *Proceedings of the Virtual Reality Annual International Symposium (VRAIS'95)*, pp. 203.

Adams, R.J., Hannaford, B. (2002). “Control law design for haptic interfaces to virtual reality”, *IEEE Transactions on Control Systems Technology*, Vol. 10, No. 1, pp. 3-13.

Adams, R.J., Hannaford, B. (1999). “Stable haptic interaction with virtual environments”, *IEEE Transactions on Robotics and Automation*, Vol. 15, No. 3, pp. 465-474.

Agus, M., Giachetti, A., Gobbetti, E., Zanetti, G., John, N.W., and Stone, R.J. (2002) "Mastoidectomy Simulation with Combined Visual and Haptic Feedback,", *Proceedings of Medicine Meets Virtual Reality Conference*, pp. 17-23.

Avila, R. S. and Sobierajski, L. M., (1996). "A Haptic Interaction Method for Volume Visualization", IEEE Proceedings of Visualization, pp. 197-204.

Bartz, D., and Guvit, O. (2000). "Haptic Navigation in Volumetric Datasets," *In Phantom User Research Symposium*, Zurich.

Basdogan, C, De S., Jung Kim, Muniyandi, M., Kim, H.,Mandayam A. Srinivasan (2004). “Haptics in Minimally Invasive Surgical Simulation and Training”, *IEEE Computer Graphics and Applications*, Vol. 24, No.2, pp. 56-64.

Basdogan, C., Srinivasan, M.A., (2002). “Haptic Rendering In Virtual Environments”, *Handbook of Virtual Environments*, Ed.: K. Stanney, pp. 117-134

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", the *Winter Annual Meeting of ASME’97*, DSC-Vol. 61, pp. 77-84, Dallas, TX, Nov. 16-21.

Chen, H., and Sun, H. (2002). "Real-time Haptic Sculpting in Virtual Volume Space," *Proc. ACM Symposium on Virtual Reality Software and Technology*) pp 81-88.

Chen, K.W., Heng, P.A., and Sun, H. (2000). "Direct Haptic Rendering of Isosurface by Intermediate Representation," *VRST: Proc. of ACM Symposium on Virtual Reality Software and Technology*, pp 188-194.

Choi, S., Tan, H., (2004). "Toward Realistic Haptic Rendering of Surface Textures", *IEEE Computer Graphics and Applications*, Vol. 24, No.2, pp. 40-47.

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*, pp. 3205-3210.

Dachille, F., Qin, H., Kaufman, A., El-sanat J., (1999). “Haptic sculpting of dynamic surfaces*”, ACM Symposium on Interactive 3D Graphics*, pp. 103-110.

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.

Eriksson, M., Flemmer, H., and Wikander, J. (2005). "A Haptic and Virtual Reality Skull Bone Surgery Simulator," *Proceedings of World Haptics*, March, Pisa, Italy.

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.

Gibson, S.F., Samosky, J., Mor, A., Fyock, C., Grimson, W.E.L., Kanade, T., Kikinis, R., Lauer, H.C., McKenzie, N., Nakajima, S., Ohkami, T., Osborne, R., and Sawad, A. (1997). "Simulating Arthroscopic Knee Surgery using Volumetric Object Representations, Real-Time Volume Rendering and Haptic Feedback," *Proceedings of CVRMed-MRCAS*, pp 369-378.

Gibson, S.F. (1995). "Beyond Volume Rendering: Visualization, Haptic Exploration and Physical Modeling of Voxel-based Objects," *MERL Technical Report 95-004*.

Gibson, S.F. (1997). "3D ChainMail: a fast algorithm for deforming volumetric objects," ***ACM** Symposium on Interactive 3D Graphics***,** pp 149-154.

Gillespie, B. Cutkosky, M., (1996). "Stable User-Specific Haptic Rendering of the Virtual Wall," *Proceedings of ASME Conference*, Vol. 58, 1996, pp. 397-406.

Gregory, A., Mascarenhas, A., Ehmann, S., Lin, M. and Manocha, D. (2001) Six Degree-of-Freedom Haptic Display of Polygonal Models, *Proc. of IEEE Visualisation Conf.*, pp. 139-146.

Ho, C., Basdogan, C. and Srinivasan, M. A. (1999) *Presence: Teleoperators and Virtual Environments*, Vol. 8, No. 5, pp. 477-491.

Ho, C., Basdogan, C., Srinivasan, M.A., 2000, "Ray-based Haptic Rendering: Interactions Between a Line Probe and 3D Objects in Virtual Environments", *International Journal of Robotics Research*, Vol. 19, No. 7, pp. 668-683.

Ikits, M., Brederson, J.D., Hansen, C., and Johnson, C. (2003). "A Constraint-Based Technique for Haptic Volume Exploration," *Proc. of IEEE Visualization Conf.*, pp. 263-269.

Iwata, H. and Noma, H. (1993) In *Proc. of IEEE Symp. on Research Frontiers in Virtual Reality*, pp. 16-23.

Johnson, D. E. and Willemsen, P. (2003). "Six Degree-of-Freedom Haptic Rendering of Complex Polygonal Models", *Proceedings of 11th Symp. on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pp. 229-235.

Kim, L., Kyrikou, A., Sukhatme, G. S. and Desbrun, M. (2002). *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2943-2948.

Klatzky, R. L., Lederman, S. J., Hamilton, C., Grindley, M. and Swendsen, R. H. (2003). *Perception \& Psychophysics*, Vol. 65, No. 4, pp. 613-631.

Körner, O., Schill, M., Wagner, C., Bender, H.J., and Männer, R. (1999). "Haptic Volume Rendering with an Intermediate Local Representation," *Proc. 1st International Workshop on Haptic Devices in Medical Applications*, pp 79-84.

Kusumoto, N., Sohmura, T., Yamada, S., Wakabayashi, K., Nakamura, T., and Yatani, H., (2006). "Applcation of virtual reality force feedback haptic device for oral implant surgery," *Clinical Oral Impl. Res.* Vol. 17, pp 708-713.

Lacroute, P., and Levoy, M. (1994). "Fast Volume Rendering Using a Shear-Warp Factorization of the viewing transformation," *ACM Siggraph*, pp. 451-458.

Laur, D., and Hanrahan, P. (1991). "Hierarchical Splatting: A progressive refinement algorithm for volume rendering," *ACM Siggraph*, pp 285-288.

Laycock, S.D., and Day, A.M. (2005). Incorporating Haptic Feedback for the Simulation of a Deformable Tool in a Rigid Scene, *Computers & Graphics*, Vol. 29, No. 3, pp. 341-351.

Laycock, S.D., Day, A.M. (2007) A Survey of Haptic Rendering Techniques, Computer Graphics Forum, Vol. 26, No. 1, pp 50-65.

Lawrence, D.A., Lee, C.D., Pao, L.Y., and Novoselov, R.Y. (2000). "Shock and Vortex Visualization using a combined visual/haptic interface," *Proc. of IEE Visualization Conf.*, pp 131-137.

Lundin, K., Ynnerman, A., and Gudmundsson, B. (2002). "Proxy-based Haptic Feedback from Volumetric Density Data," *EuroHaptics*, pp 104-109.

Maneewarn, T., Storti, D. W., Hannaford, B. and Ganter, M. A. (1999). "Haptic Rendering for Internal Content of an Implicit Object", *Proc. ASME Winter Annual Meeting Haptic Symposium*, Nashville, TN.

Mark, W. R., Randolph, S. C., Finch, M., Verth, J. M. V. and II, R. M. T. (1996). "Adding Force Feedback to Graphics Systems: Issues and Solutions", *ACM Siggraph*, Louisiana, pp. 447-452.

McNeely, W., Puterbaugh, K., and Troy, J. (1999). Six degree-of-freedom haptic rendering using voxel sampling. *Proc. of ACM Siggraph*, pp. 401–408.

Miller, B.E., Colgate, J.E., and Freeman, R.A. (2000). "Guaranteed stability of haptic systems with nonlinear virtual environments", *IEEE Transactions on Robotics and Automation*, Vol. 16, No. 6, pp. 712-719.

Minsky, M., Ouh-young, M., Steele, O., Brooks, F. P. and Behensky, M. (1990). Feeling and Seeing in Force Display", *ACM Siggraph*, Vol. 24(2), pp. 235-243.

Morgenbesser, H.B., Srinivasan, M.A. (1996). "Force Shading for Haptic Shape Perception", *Proceedings of the ASME Dynamic Systems and Control Division*, Vol. 58, 407-412.

Nelson, D. D., Johnson, D. E., and Cohen, E. (1999). "Haptic rendering of surface-to-surface sculpted model interaction", *Proc. of ASME Dynamic Systems and Control Division.*

Otaduy, M. A., Jain, N., Sud, A. and Lin, M. C. (2004). "Haptic display of interaction between textured models", *Proc. of IEEE Visualization Conference*Austin, TX, pp. 297-304.

Otaduy, M. A., and Lin, M. C. 2005. Stable and responsive six-degree-of-freedom haptic manipulation using implicit integration. *Proc. of World Haptics Conference.*

Palmerius, K.L. (2007). "Fast and high Precision Volume Haptics," to appear in *Proc. World Haptics Conference*.

Patoglu, V., Gillespie R.B. (2005). "A closest point algorithm for parametric surfaces with global uniform asymptotic stability", *Proceedings of IEEE Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*.

Patoglu, V., Gillespie R.B. (2004). "Haptic Rendering of Parametric Surfaces Using a Feedback Stabilized Extremal Distance Tracking Algorithm", *Proceedings of IEEE Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*.

Petersik, A., Pflesser, B., Tiede, U., Höhne, K.H., and Leuwer, R. (2002). "Realistic Haptic Volume Interaction for Petrous Bone Surgery Simulation," *Proc. of CARS Conf.*.

Ruspini, D.C., Kolarov, K., Khatib O. (1997). The Haptic Display of Complex Graphical Environments. *ACM (Proceedings of SIGGRAPH)*, July, pp. 345-352.

Salcudean, S. E. and Vlaar, T. D. (1994). On the Emulation of Stiff Walls and Static Friction with a Magnetically Levitated Input/Output Device. *Proceedings of ASME*, DSC, Vol. 55, No.1, pp. 303 - 309.

Salisbury, K., Brock, D., Massie, T., Swarup, N. and Zilles, C. (1995). "Haptic rendering: programming touch interaction with virtual objects", *Proc. of the Symp. on Interactive 3D Graphics*, pp. 123-130.

Salisbury, J.K., and Tarr, C., (1997). "Haptic Rendering of Surfaces Defined by Implicit Functions", *Proceedings of the ASME*, DSC-61, pp. 61-67.

Salisbury, J. K., Srinivasan, M. A. (1997). "Phantom-Based Haptic Interaction with Virtual Objects", *IEEE Computer Graphics and Applications*, Vol. 17, No. 5, pp. 6-10.

Salisbury, K., Conti, F., Barbagli, F. (2004). "Haptic Rendering: Introductory Concepts", *IEEE Computer Graphics and Applications*, Vol. 24, No. 2, pp. 24-32.

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.

Siira, J., Pai D. K. (1996). Haptic Texturing - A Stochastic Approach. *Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis, Minnesota, pp. 557-562.

Srinivasan, M.A., and Basdogan, C. (1997). Haptics in Virtual Environments: Taxonomy, Research Status, and Challenges. Computers and Graphics, Vol. 21, No. 4, pp. 393 - 404.

Thompson, T. V., Johnson, D. E. and Cohen, E. (1997). "Direct Haptic Rendering Of Sculptured Models", *Proc. of Symp. on Interactive 3D Graphics*, pp. 167-176.

Zilles, C. B. and Salisbury, J. K. (1995). "A constraint-based god-object method for haptic display", *Proc. of the IEEE Conference on Intelligent Robots and Systems*, pp. 146-151.