Six Degree-of-Freedom Haptic Rendering Using Voxel Sampling

William A. McNeely **Kevin D. Puterbaugh** **James J. Troy**

The Boeing Company*

Abstract

A simple, fast, and approximate voxel-based approach to 6-DOF haptic rendering is presented. It can reliably sustain a 1000 Hz haptic refresh rate without resorting to asynchronous physics and haptic rendering loops. It enables the manipulation of a modestly complex rigid object within an arbitrarily complex environment of static rigid objects. It renders a short-range force field surrounding the static objects, which repels the manipulated object and strives to maintain a voxel-scale minimum separation distance that is known to preclude exact surface interpenetration. Force discontinuities arising from the use of a simple penalty force model are mitigated by a dynamic simulation based on virtual coupling. A generalization of octree improves voxel memory efficiency. In a preliminary implementation, a commercially available 6-DOF haptic prototype device is driven at a constant 1000 Hz haptic refresh rate from one dedicated haptic processor, with a separate processor for graphics. This system yields stable and convincing force feedback for a wide range of user controlled motion inside a large, complex virtual environment, with very few surface interpenetration events. This level of performance appears suited to applications such as certain maintenance and assembly task simulations that can tolerate voxel-scale minimum separation distances.

CR Categories and Subject Descriptors: H.5.2 [User Interfaces]: Haptic I/O, I.3.5 [Computational Geometry and Object Modeling]: Physically Based Modeling.

Additional Keywords: force feedback, voxel representations, virtual environments.

1. INTRODUCTION

The problem of simulating real-world engineering tasks — for example, objectives like design-for-assembly and design-for-maintenance — has been exacerbated by the modern transition from physical mockup to virtual mockup. Physical mockup provides natural surface constraints that prevent tools and parts from interpenetrating, whereas virtual mockup requires the user to satisfy such constraints by receiving collision cues and making appropriate body postural adjustments, which is usually tedious and may yield dubious results. In order to emulate the natural surface constraint satisfaction of physical mockup, one must introduce force feedback into virtual mockup. Doing so shifts the burden of physical constraint satisfaction onto a haptic subsystem, and the user becomes free to concentrate on higher-level problems such as path planning and engineering rule satisfaction.

Tool and part manipulation inherently requires six degree-of-freedom (6-DOF) haptics, since extended objects are free to move in three translational and three rotational directions. Affordable high-bandwidth 6-DOF devices are becoming available, but 6-DOF haptic rendering remains an outstanding problem. It is considerably more difficult than 3-DOF point-contact haptic rendering. One can compare haptics with collision detection, since they share some technical similarity. Real-time collision detection is a challenging problem [13], but 6-DOF haptics adds stringent new requirements such as:

- Detect all surface contact (or proximity, for a force field), instead of stopping at the first evidence of it.
- Calculate a reaction force and torque at every point or extended region of contact/proximity.
- Reliably maintain a 1000 Hz refresh rate, independent of position and orientation of the manipulated object.
- Control geometry driven haptic instabilities, such as forcing an object into a narrow wedge-shaped cavity.

To address the needs of our targeted engineering applications, we adopt the following additional goals:

- Minimize the interpenetration of exact surface representations.
- Handle complex static scenes, e.g., those containing several hundred thousand triangles, with reasonable memory efficiency.
- The haptic rendering algorithm should parallelize easily.

Furthermore, we accept the limitation of voxel-scale accuracy. For example, a common engineering rule is to design at least 0.5 inch clearance into part removal paths, whenever possible, in order to accommodate tool access and human grasping and to serve as a cushion against assembly tolerance buildup.

We describe an approach that formally meets most of these requirements. It demonstrates the ability to drive a commercially available 6-DOF prototype device at a reliable 1000 Hz haptic refresh rate without the aid of asynchronous physics and haptic rendering loops. It supports the manipulation of a single rigid object within an arbitrarily rich environment of static rigid objects by rendering a half-voxel-deep force field that surrounds the static objects and serves to block potential interpenetration of the exact surface representations, as described in section 4.2. Given a predetermined spatial accuracy (i.e., voxel size), rendering performance depends linearly on the total exposed surface area of the manipulated object. There is also a relatively minor dependence on the instantaneous amount of contact/proximity, with a worst-case performance (e.g., maximum contact/proximity) of about half that of the best-case performance.

* Bill McNeely, The Boeing Company, P.O. Box 3707, M/S 7L-43, Seattle, WA 98124, (bill.mcneely@boeing.com)

SIGGRAPH 99, Los Angeles, CA USA

Our approach is distinguished primarily by its high haptic rendering speed, which is derived primarily from:

- A simple penalty force scheme called the tangent-plane force model, explained in section 3.
- A fixed-depth voxel tree, explained in section 4.3.
- A voxel map that collectively represents all static objects, explained in section 4.4.

Although the simplicity of our force model is critically important to performance, it is so simple that it generates force magnitude discontinuities (but not force direction discontinuities), especially under sliding motion. In 3-DOF point-contact haptics, force discontinuities can be devastating to force quality and stability, but under our 6-DOF approach there is a stochastic effect that lessens their impact. However, it proved necessary to introduce various measures to explicitly enhance force quality and stability, such as:

- A single-body dynamic model based on virtual coupling
- Pre-contact braking forces

All such measures are explained in section 5.

Data storage is often a secondary consideration in haptics work, because it is tempting to trade memory efficiency for higher performance. However, voxels are so relatively inefficient as geometric modeling elements that we improve their memory efficiency by generalizing the octree method, as explained in section 4.3.

2. PREVIOUS WORK

Although largely the result of unpublished work, there are numerous examples of 6-DOF haptic rendering for scenarios containing a very limited number of geometrically well behaved virtual objects, for example [6,7,24]. Our approach differs from this work primarily in its ability to render considerably more complex 6-DOF scenarios with no formal constraints on object shape, although at reduced accuracy.

Our approach includes a collision detection technique based on probing a voxelized environment with surface point samples. Voxel-based methods have been applied to non-haptic collision detection [12,15,16] and to 3-DOF haptics [3,18]. Sclaroff and Pentland [22] apply surface point sampling to implicit surfaces.

Intermediate representations for haptics were suggested by Adachi et al. [1], and have been subsequently elaborated [17]. This involves using a simple haptics proxy that approximates the exact scene and is simple enough to update the forces at the required high refresh rate, while a slower but more exact collision detection and/or dynamic simulation runs asynchronously and updates the proxy's parameters. Our work differs by tightly integrating collision detection, the force model, and the dynamic model into a single loop that updates forces directly at 1000 Hz.

There has been much work in multibody dynamic simulation for physically based modeling, for example [4,23]. Mirtich and Canny [19] track the contacts found from an iterative collision detection method and use this information to generate constant-size impulses. In general, such work is characterized by its emphasis on accuracy over rendering performance, and consequently it relies on methodology such as exact-surface collision detection and simultaneous surface constraint satisfaction, which currently fall far short of 6-DOF haptics performance requirements.

Our dynamic model adopts the practice of using an artificial coupling between the haptic display and virtual environment, as originally proposed by Colgate et al. [10] and recently elaborated by Adams and Hannaford [2]. We also adopt a version of the "god object" concept suggested by Zilles and Salisbury [25] and others [21], generalized to 6-DOF and modified to use penalty forces that only approximately satisfy surface constraints. In addition, we use the concept of pre-contact braking force suggested by Clover [9].

Hierarchical techniques, such as employed by Gottschalk [13], can be used to alleviate convex-hull bounding box limitations for objects in very close proximity by recursively generating a tree of bounding volumes around finer features of the object. While this technique speeds collision detection, it also introduces indeterminacy in the cycle rate due to the varying cost of traversing the tree structure to an unknown depth to check each colliding polygon against object polygons. Cycle-rate should not only be fast but should also have a rate that is as constant as possible.

Temporal and spatial coherence can also be exploited [4,5,8] by assuming that objects move only slightly within each time step, thus allowing extrapolation from the previous state of the system. The number of polygon tests carried out at each time step is effectively reduced, increasing cycle-rate at the cost of introducing indeterminacy. With certain configurations or motions of objects, however, there are often noticeable drops in performance — a situation which is unacceptable in a real-time simulation.

3. TANGENT-PLANE FORCE MODEL

In our tangent-plane force model, dynamic objects are represented by a set of surface point samples, plus associated inward pointing surface normals, collectively called a point shell. During each haptic update the dynamic object's motion transformation is applied to every point of the point shell. The environment of static objects is collectively represented by a single spatial occupancy map called a voxmap, which is illustrated in Figure 1. Each haptically rendered frame involves sampling the voxmap at every point of the point shell.

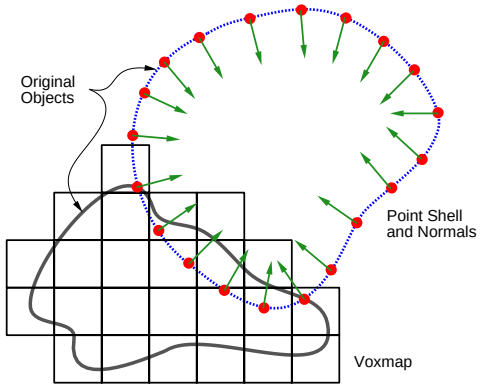

Figure 1. Voxmap colliding with point shell.

When a point interpenetrates a voxel (assumed for now to be a surface voxel) as shown in Figure 2, a depth of interpenetration is calculated as the distance d from the point to a plane within the voxel called the tangent plane.

The tangent plane is dynamically constructed to pass through the voxel's center point and to have the same normal as the point's associated normal. If the point has not penetrated below that plane (i.e., closer to the interior of the static object), then $\boldsymbol{d}$ is zero. Force is simply proportional to $\boldsymbol{d}$ by Hooke's law ($\boldsymbol{F} = K_{ff}\boldsymbol{d}$). We call K_{ff} the "force field stiffness," since the voxel represents a half-

voxel-deep force field. The net force and torque acting on the dynamic object is obtained as the sum of all force/torque contributions from such point-voxel intersections.

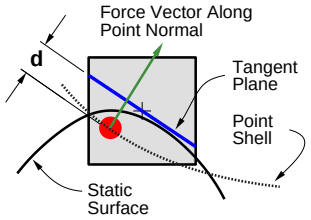

Figure 2. Tangent-plane force model.

The tangent-plane force model was inspired by the fact that the surfaces of contacting objects are tangent at an osculation point. It is important that the force takes its direction from a precomputed surface normal of the dynamic object. This proves to be considerably faster than the common practice of dynamically computing it from the static object's surface, or in the case of a force field, dynamically taking the gradient of a potential field.

One can see that this simple model has discontinuities in force magnitude when a point crosses a voxel boundary, for example, under sliding motion. Section 5 describes how discontinuities can be mitigated for haptic purposes.

4. VOXEL DATA STRUCTURES

This section outlines the creation and usage of voxel-based data structures that are required under our approach. Exact (polygonal) surface penetration and memory usage will also be discussed.

4.1 Voxmap and Point Shell

One begins by selecting a global voxel size, *s*, that meets the virtual scenario's requirements for accuracy and performance. The performance aspect is that the force model requires traversing a set of point samples, and *s* determines the number of such points. Consider a solid object such as the teapot in Figure 3(a). It partitions space into regions of free space, object surface, and object interior. Now tile this space into a volume occupancy map, or voxmap, as in Figure 3(b). The collection of center points of all surface voxels constitutes the point shell needed by the tangent-plane force model, as in Figure 3(c).

Figure 3. Teapot: (a) polygonal model, (b) voxel model, (c) point shell model.

This method for creating the point shell is not optimal, but it is convenient. Its accuracy may be improved by choosing points that lie on the exact geometrical representation.

Each voxel is allocated two bits of memory that designate it as a free space, interior, surface, or proximity voxel. The 2-bit voxel types are defined in Table 1 and illustrated by an example in Figure 4.

A neighbor voxel is defined as sharing a vertex, edge, or face with the subject voxel. Each voxel has 26 neighbors. It is important that each static object be voxelized in its final position and orientation in the world frame, because such transformations cause its voxelized representation to change shape slightly.

Table 1. Voxel types (2-bit)

Value	Voxel type	Description
0	Free space	Encloses only free-space volumes
1	Interior	Encloses only interior volumes
2	Surface	Encloses a mix of free-space, surface, and interior volumes
3	Proximity	Free-space neighbor of a surface voxel

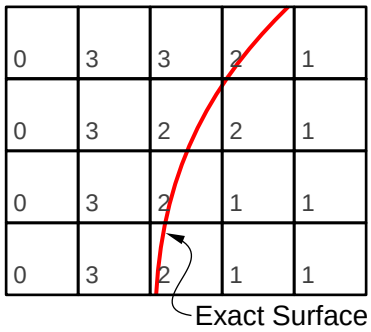

Figure 4. Assignment of 2-bit voxel values.

By the nature of 3D scan conversion, voxmaps are insensitive to surface imperfections such as gaps or cracks that are smaller than the voxel width. However, identifying the interior of a voxmap can be difficult. We adopt the practice of (1) scan-converting to create surface voxels, (2) identifying free-space voxels by propagating the voxelized walls of the object's bounding box inward until surface voxels are encountered, and (3) declaring all other voxels to be interior voxels. This ensures that objects with open surfaces will be voxelized instead of "leaking" and filling all voxels.

4.2 Avoiding Exact Surface Interpenetration

In the tangent-plane force model shown in Figure 2, the exact surfaces of colliding objects are allowed to interpenetrate by voxel-scale distances during a point-voxel intersection. While this may be acceptable for some applications, we seek instead to preclude exact-surface interpenetration. We do this by offsetting the force field outward away from the surface by two voxel layers, as shown in Figure 5. (In this figure, the rotated boxes represent the surface voxels associated with the points of a pointshell, viewed as surface bounding volumes.) The offset force layer then serves to maintain a minimum object separation that provably precludes exact-surface interpenetration.

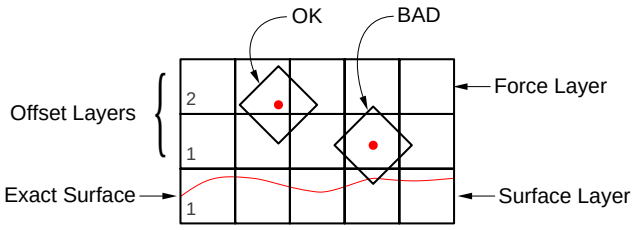

Figure 5. Criterion for exact-surface interpenetration.

The voxel legend described by Table 1 and Figure 4 is correspondingly redefined so that "surface" and "value 2" now refer to the offset force-layer voxels instead of geometric surface voxels, and similarly for the other voxel types. (Offset proximity voxels and free-space voxels are omitted from Figure 5, but they would occupy additional layers at the top of the figure.)

Force-layer offsetting is implemented as a final step of voxelization, in which the geometric surface voxel layer is grown outward by a process of promoting proximity voxels to surface values and demoting original surface voxels to interior values. This process is repeated to achieve the desired two-layer offset. (If voxels were allocated more than two bits, it would not be necessary to "recycle" voxel values in this manner, and there are other advantages to wider voxels that we are beginning to explore.)

Force-layer offsetting also serves to prevent any spike-like feature in the static object from generating a linear column of voxels that the point shell could completely fail to penetrate for certain orientations of the dynamic object. The force layer has no such features, because voxel values are propagated to 26 connected neighbors during the offsetting process.

4.3 Voxel Tree

A natural next step is to impose an octree organization on the voxels for the sake of memory efficiency and scalability. However, the need for a consistently fast haptic refresh rate is at odds with the variability in the tree traversal time. To address this, we have devised a hierarchy that represents a compromise between memory efficiency and haptic rendering performance. It is a generalization of octree with a tree depth that is limited to three levels, explained as follows.

At each level of the tree, the cubical volume of space is divided into 2^{3N} sub-volumes, where N is a positive integer. (N is unity for an octree.) We have discovered that the most memory-efficient value for N may be at higher values, depending on the sparseness of the geometry. Figure 6 illustrates a study of the total memory consumed by a 2^{3N}-tree as a function of N for geometry that is typical to our work. It has a minimum at N=3, which might be called a 512-tree.

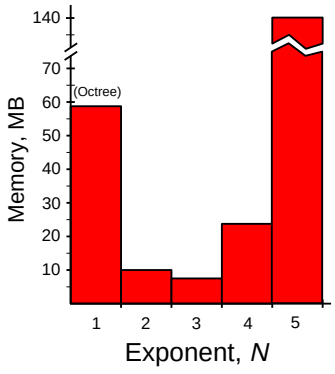

Figure 6. Memory usage of 2^{3N} tree as a function of N.

We further limit tree depth by fixing both the minimum and maximum dimensions of the bounding volumes in the tree. The minimum dimension is the size of voxels at the leaf level, and the maximum dimension is given implicitly by creating only three levels above the leaf level. The minimum-size requirement means that smaller features may not be adequately represented, but we fundamentally accept a global accuracy limitation, analogous to the practice of accepting a fixed tessellation error in polygonal surface representations. The maximum-size requirement impacts memory efficiency and scalability, because one must cover all remaining space with the largest-size bounding volumes. However, these effects are mitigated by the use of 2^{3N}-tree, since for a fixed number of levels, higher values of N increase the dynamic range of the bounding volume dimensions.

4.4 Merged Scene Voxmap

Our approach is limited to the case of a single dynamic rigid object interacting with an arbitrarily rich environment of static rigid objects. If it were necessary to separately calculate the interaction force for each of N static objects, then the computing burden would grow linearly with N. However, there is no inherent need to separately compute such interactions on a pairwise basis. For example, there is no need to identify the type of a contacted object in order to apply different material properties, since all static objects are treated as rigid. Furthermore, under our force-field approach, objects are never actually contacted in the sense of undergoing surface intersections. Therefore, we merge all static-object voxel representations together as if they were a single static object, applying straightforward precedence rules to merged voxel values and recalculating a voxel tree for the voxmap.

5. DYNAMIC MODEL

For the dynamic model, we use an impedance approach, in which user motion is sensed and a force/torque pair is produced. We further adopt what is called the "virtual coupler" scheme, which connects the user's haptic motions with the motions of the dynamic object through a virtual spring and damper. This is a well known method for enhancing haptic stability [2].

To solve for the motion of the dynamic object, we perform a numerical integration of the Newton-Euler equation, using a constant time step Δt corresponding to the time between force updates, e.g., Δt =1 msec for 1000 Hz haptic refresh rate. We also must assign a mass m to the dynamic object equal to the apparent mass for the dynamic object that we want to feel at the haptic handle (in addition to the haptic device's intrinsic friction and inertia, and assuming that its forces are not yet saturated). The net force and torque on the dynamic object is the sum of contributions from the spring-damper system, explained in section 5.1; stiffness considerations, explained in section 5.2; and the pre-contact braking force, explained in section 5.3.

5.1 A 6-DOF Spring-Damper System

Conceptually, a copy of the haptic handle is placed in the virtual scene and is coupled to the dynamic object through a spring-damper connection, as shown in Figure 7.

The real haptic handle controls the position and orientation of its virtual counterpart. This influences the spring's displacement, which generates a virtual force/torque on the dynamic object and an opposite force/torque on the real haptic handle. Spring displacement also includes rotational motion, as shown in Figure 7 by the spiral at the center of the dynamic object (suggestive of a clock mainspring). Spring force is proportional to displacement, while spring torque is proportional to the angle of rotation from an equivalent-angle analysis and directed along an equivalent axis of rotation [11].

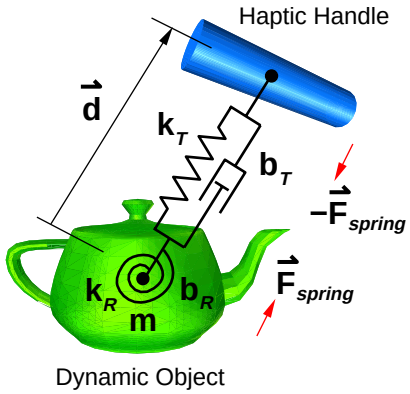

Figure 7. Dynamic model based on virtual coupling.

This 6-DOF spring makes the dynamic object tend to acquire the same position and orientation of the virtual haptic handle, assuming that the two objects are initially registered in some manner, e.g., with the center of the handle located at the dynamic object's center of mass and the handle's main axis aligned with one of the dynamic object's principal axes. The virtual object is assigned mass properties, which are reflected at the haptic interface as apparent mass that is added to the haptic device's intrinsic inertia. We operated at a small reflected mass of 12 g. The force and torque equations used here are:

$$\boldsymbol{F}_{spring} = k_T \boldsymbol{d} - b_T \boldsymbol{v}$$

$$\hat{\tau}_{spring} = k_R \hat{\theta} - b_R \vec{\omega}$$

where

k_T, b_T = spring translational stiffness and viscosity

k_R, b_R = spring rotational stiffness and viscosity

$\hat{\theta}$ = equivalent-axis angle (including axis direction)

$\boldsymbol{v}$, $\vec{\omega}$ = dynamic object's relative linear and angular velocity.

Spring stiffness is set to a reasonably high value that is still comfortably consistent with stable numerical behavior at the known time sampling rate. Stiffness and viscosity are straightforwardly related to obtain critically damped behavior. A limitation of this simple formalism is that it is only valid for a dynamic object having equal moments of inertia in every direction, such as a sphere of uniform mass density. Since we were not interested in reflected moments of inertia, and indeed sought to minimize them, this was an acceptable limitation. It represents an implicit constraint on the virtual object's mass density distribution but not on its geometrical shape.

5.2 Virtual Stiffness Considerations

When the virtual object is in resting contact with the half-voxel-deep force field described by stiffness K_{ff}, we want to prevent the user from stretching the spring so far as to overcome the force field and drag the dynamic object through it. The spring force is clamped to its value at a displacement of *s/2*, where *s* is the voxel size. In the worst case, this contact force is entirely due to a single point-voxel interaction, which therefore determines an upper limit on the spring force. This can be viewed as a modification of the god-object concept [25], in which the god-object is allowed to penetrate a surface by up to a half voxel instead of being analytically constrained to that surface.

Whenever many point-voxel intersections occur simultaneously, the net stiffness may become so large as to provoke haptic instabilities associated with fixed-time-step numerical integration. To cope with this problem, we replace the vector sum of all point-voxel forces by their average, i.e., divide the total force by the current number of point-voxel intersections, *N*. This introduces force discontinuities as *N* varies with time, especially for small values of *N*, which degrades haptic stability. We mitigate this side effect by deferring the averaging process until $N = 10$ is reached:

$$F_{Net} = F_{Total} \quad \text{if} \quad N < 10$$

$$F_{Net} = \frac{F_{Total}}{N/10} \quad \text{if} \quad N \geq 10$$

and similarly for torque. K_{ff} is adjusted to assure reasonably stable numerical integration for the fixed time step and at least 10 simultaneous point-voxel intersections. While this heuristic leads to relatively satisfactory results, we are investigating a hybrid of constraint-based and penalty-based approaches that formally address both the high-stiffness problem and its dual of low stiffness but high mechanical advantage. Forcing an object into a narrow wedge-shaped cavity is an example of the latter problem.

Dynamic simulation is subject to the well studied problem of non-passivity, which might be defined as the unintended generation of excessive virtual energy [2,10]. In a haptic system, non-passivity manifests itself as distracting forces and motions (notably, vibrations) with no apparent basis in the virtual scenario. Non-passivity is inherent in the use of time-sampled penalty forces and in the force discontinuity that is likely to occur whenever a point crosses a voxel boundary. Another potential source of non-passivity is insufficient physical damping in the haptic device [10]. Even a relatively passive dynamic simulation may become highly non-passive when placed in closed-loop interaction with a haptic device, depending on various details of the haptic device's design, its current kinematic posture, and even the user's motion behavior.

The most direct way to control non-passivity is to operate at the highest possible force-torque update rate supported by the haptic device, which for our work was the relatively high value of 1000 Hz. We also investigated the technique of computationally detecting and dissipating excessive virtual energy. While this had some success, it was eventually replaced by the simpler technique of empirically determining the largest value of K_{ff} consistent with stable operation over the entire workspace of the haptic device. As a further refinement, we discovered some residual instability in the dynamic object when it lies in free space. Whenever that occurs, therefore, we apply zero force and torque to the haptic device (overriding any non-zero spring values). A free-space configuration is trivially detected as every point of the dynamic object intersecting a free-space voxel of the environment.

5.3 Pre-Contact Braking Force

The treatment of spring-force clamping in section 5.2 ignored the fact that the dynamic object's momentum may induce deeper instantaneous point-voxel penetration than is possible under resting contact, thereby overcoming the force field. Currently, we do not attempt to avoid this outcome in every instance. Instead, we generate a force in the proximity voxel layer that acts to reduce the point's velocity, called the pre-contact braking force. In order to avoid a surface stickiness effect, the force must only act when the point is approaching contact, not receding from a prior contact. To determine whether the point is approaching or receding, consult its

associated inward-pointing surface normal, $\hat{n}_i$, and then calculate the force:

$$\boldsymbol{F}_i = -b\boldsymbol{v}_i(-\hat{n}_i \cdot \hat{v}_i), \text{ if } \hat{n}_i \cdot \hat{v}_i < 0$$

$$\boldsymbol{F}_i = 0, \text{ if } \hat{n}_i \cdot \hat{v}_i \geq 0$$

where b is a "braking viscosity," $\boldsymbol{v}_i$ is the velocity of the i^{th} point in the point shell, and $\hat{v}_i$ is a unit vector along $\boldsymbol{v}_i$.

As a simple heuristic, therefore, adjust b so as to dissipate the object's translational kinetic energy along the direction of approaching contact within one haptic cycle:

$$b = \frac{(\frac{1}{2}mv^2)/\Delta t}{\boldsymbol{v} \cdot \Sigma_i \boldsymbol{v}_i(-\hat{n}_i \cdot \hat{v}_i)}$$

where m and $\boldsymbol{v}$ are the dynamic object's mass and velocity component along $\Sigma \boldsymbol{F}_i$, respectively, and the sum over i is understood to traverse only points for which $\hat{n}_i \cdot \hat{v}_i < 0$.

We have not yet implemented a braking torque. Calculating this type of torque would be similar in form to the translational braking viscosity equation above.

A weakness of the braking technique is that an individual point's velocity may become so large that the point skips over the proximity voxel in a single haptic cycle, or even worse, over all voxels of a thin object. We call this the "tunnelling problem." This is particularly likely to happen for points of a long dynamic object that is rotated with sufficient angular velocity. One possible solution is to constrain the dynamic object's translational and angular velocities such that no point's velocity ever exceeds $s/\Delta t$.

6. RESULTS

The system configuration for our preliminary implementation is illustrated in Figure 8. Haptic rendering is performed on a dedicated haptics processor, which asserts updated force and torque information to the haptic device and reads position and orientation of the haptic handle in a closed loop running at 1000 Hz.

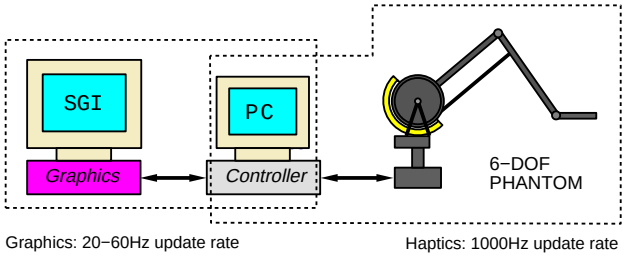

Figure 8. System Configuration.

In a separate asynchronous open loop, the haptics processor transmits UDP packets containing position and orientation information to a dedicated graphics processor, which renders the updated scene at about 20 Hz. This section provides more details on the system components and presents some preliminary results.

6.1 Haptics Device

We used a desk-mounted system called the PHANTOM™ Premium 6-DOF Prototype (shown in Figure 9), made by SensAble Technologies, Inc. This system includes the mechanism, its power electronics, a PCI interface card, and the GHOST® Software Developer's Kit (SDK). Force feedback in three translational degrees-of-freedom is provided by a vertical 2-link planar structure, with a third orthogonal rotational axis at the base. Cable transmission actuators drive linkages from the base. Its peak force is 22 N and the nominal positioning resolution is 0.025 mm at the end effector. The translational range of motion is about 42×59×82 cm, approximating the natural range of motion of the entire human arm. Torque feedback in three rotational degrees of freedom is provided by a powered gimbal mechanism that provides torques in yaw, pitch, and roll directions. Its peak torque is 0.67 Nm and the nominal resolution is 0.013˚ in each axis. The rotational range of motion is 330˚ in both yaw and roll, and 220˚ in pitch.

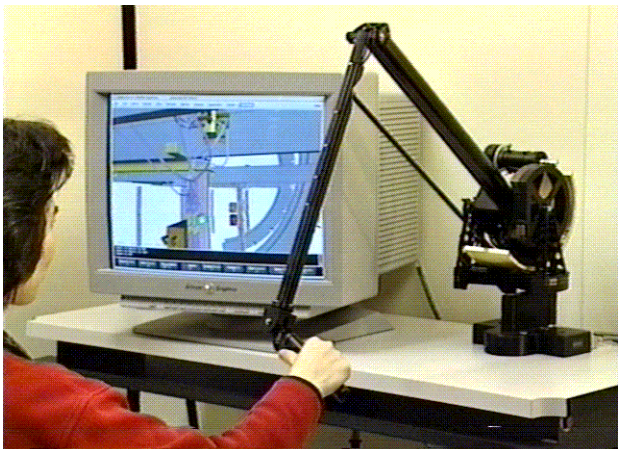

Figure 9. User with the 6-DOF haptic device.

Low-level interactions with the PCI interface card are handled by the PHANTOM device drivers provided with the system. The GHOST SDK transparently provides real-time motion control, including the use of a proprietary mechanism that guarantees a 1 kHz servo rate. A kinematic model that deals with conversions between joint space and Cartesian space, and dynamics algorithms that optimize the feel by compensating for device dynamics. Although the GHOST SDK supports numerous high-level interactions with the system, our usage is currently limited to (1) querying for global position and orientation of the end effector as a 4×4 homogeneous transformation matrix and (2) asserting the desired global force and torque.

6.2 Haptics and Graphics Processing

The dedicated haptics processor of our prototype system was a 350 MHz Pentium® II CPU with 128 MB of RAM running Windows NT®. The functions of voxelization, voxel-sampling, and force generation were provided by Boeing developed software known as Voxmap PointShell™, which implements the approach presented in this paper. Voxmap PointShell is interfaced with the GHOST SDK, which manages the 1 kHz servo loop. Within this loop, the haptic handle's position and velocity information is received, a haptic frame is rendered, and updated force and torque information is sent to the device. GHOST monitors the time consumption of each loop and interrupts operation whenever a 1 kHz servo loop constraint is violated. Outside the servo loop, a separate, asynchronous loop samples the transformation matrices for the dynamic object and haptic handle, and sends them via UDP to a dedicated graphics processor.

Our dedicated graphics processor was an SGI Octane™ with one 250 MHz R10000 processor, 256 MB of RAM, and SI graphics. For visualization we use FlyThru®, a proprietary high-performance visualization system. This system was first used to virtually

"preassemble" the Boeing 777 and is now employed on commercial, military, and space programs throughout Boeing. FlyThru can maintain a frame rate of ~20 Hz, independent of the amount of static geometry. This is achieved by rendering the static geometry once to the color and Z-buffers, then reusing those images for subsequent frames [20]. This visualization scheme provided smooth motion with no noticeable lag.

One disadvantage of using two separate computers is that setup and usage tend to be cumbersome. In light of this, we have also implemented our approach on an Octane with two processors — one used strictly for haptics and the other for graphics.

6.3 Virtual Scenario

The static environment of our virtual scenario consisted of simulated aircraft geometry, with beams, tubes, wires, etc., voxelized at 5 mm resolution. Its polyhedral representation contains 593,409 polygons. Its FlyThru representation consumed 26 MB of memory, and its voxelized representation consumed 21 MB. Voxelization time on a 250 MHz SGI Octane was 70 sec. A closeup shot of a dynamic object (a teapot) maneuvering through a portion of this environment is shown in Figure 10.

The dynamic object for much of our testing was a small teapot (75 mm from spout to handle), logically representing a small tool or part, which when voxelized at 5 mm resolution yielded 380 points in its pointshell for the PC haptics processor. The dedicated haptics processor of the two-processor Octane system was able to achieve a maximum of 600 points for the same object.

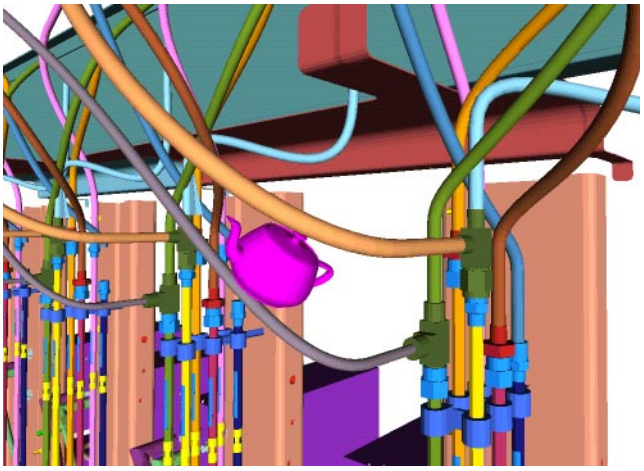

Figure 10. Dynamic object in the test environment.

6.4 Preliminary Test Results

We haptically rendered the motion of the teapot through the simulated aircraft geometry, paying particular attention to motion behavior and quality of force feedback. We evaluated the feeling of free space as well as resting and sliding contact (with the force field). In an attempt to explore the system's limits, we sought to induce haptic instabilities and exact-surface interpenetrations by trapping the teapot in congested areas and by staging high-speed collisions.

Subjectively, the observed free-space behavior was indistinguishable from power-off operation, for translational as well as rotational motion. Sliding behavior on a flat or slowly curving surface was notably smooth. A relatively slight surface roughness was felt when sliding in contact with two surfaces. Torques were clearly felt. We were able to move the teapot easily through congested areas where combinations of rotation and translation were required to find a path through the area, similar to path planning for maintenance access.

Throughout such investigation, a 1 kHz update requirement was maintained. We were unable to cause the teapot to pass completely through any of the environment surfaces, including relatively thin ones, even at maximum collision speed. There were remarkably few potential exact-surface interpenetration events. One natural metric is the ratio of penetration to collision events (PR) defined as the number of haptic frames registering one or more potential exact-surface penetrations divided by the number of haptic frames registering contact with the force-field layer (including penetration events).

We evaluated the benefit of the pre-contact braking force by selectively disabling it and re-measuring PR. The effect of this was fewer exact-surface penetrations, as shown in Table 2.

Table 2. Penetration ratio

Test	Braking	Penetrations	Contacts	PR
1	No	70	69,000	1.0×10^{-3}
2	Yes	6	108,000	6×10^{-5}

All such work was done with the haptic device limited to 15 N force and 0.1 Nm torque. At these limits we found the device to be stable for every possible type of motion.

7. CONCLUSIONS AND FUTURE WORK

The voxel-based approach to haptic rendering presented here enables 6-DOF manipulation of a modestly sized rigid object within an arbitrarily complex environment of static objects. The size of the moving object (i.e., the number of points in the point shell) is limited by the processor speed, while the size of the static environment is limited by memory. A force model was described in which the interaction of the moving object's surface normals with the static voxmap was used to create haptic forces and torques. Results of testing an implementation of our approach on a 6-DOF haptic device showed that the performance appears to be acceptable for maintenance and assembly task simulations, provided that the task can tolerate voxel level accuracy.

It is apparent to us that we are just beginning to discover all the potential uses for the voxmap sampling method in haptics and other fields. Our primary focus will be to enhance the performance of the system for use in complex environments.

The voxel sampling method can be easily parallelized, using clones of the static environment and cyclic decomposition of the dynamic object's pointshell. We intend to take advantage of this by investigating parallel computing environments, specifically low-latency cluster computing. This will allow haptic simulation of larger and more complex dynamic objects.

Another area of interest that we are pursuing involves using wider-bit-width voxel types (4-bit, 8-bit, etc.). This enhancement will allow for an extended force field range to model compliance when simulating varying material types.

We also intend to continue investigating solutions to problematic situations, like the wedge problem and tunnelling (moving through a thin object without detecting collision), as well as further reducing non-passivity.

Acknowledgments

The authors express their thanks to colleagues Karel Zikan for the idea of voxel sampling, Jeff A. Heisserman for the idea of normal-aligned force direction, Robert A. Perry for creating simulated aircraft geometry, and Elaine Chen of SensAble Technologies, Inc. for literature research and technical information about the PHANTOM device and GHOST software support.

References

[1] Adachi, T., Kumano, T., Ogino, K., "Intermediate Representations for Stiff Virtual Objects," *Proc. IEEE Virtual Reality Annual Intl. Symposium*, pp. 203-210, 1995.

[2] Adams, R.J. and Hannaford, B., "A Two-Port Framework for the Design of Unconditionally Stable Haptic Interfaces," *Proc. IROS*, Anaheim CA, 1998.

[3] Avila, R.S. and Sobierajski, L.M., "A Haptic Interaction Method for Volume Visualization," *Proc. Visualization'96*, pp. 197-204, Oct. 1996.

[4] Baraff, D., "Curved Surfaces and Coherence for Non-Penetrating Rigid Body Simulation," *Computer Graphics (proc. SIGGRAPH 90)*, vol 24, no. 4, pp. 19-28, Aug. 1990.

[5] Baraff, D., "Fast Contact Force Computation for Nonpenetrating Rigid Bodies," *Computer Graphics (proc. SIGGRAPH 94)*, pp. 23-42, July 1994.

[6] Berkelman, P.J. and Hollis, R.L., "Dynamic performance of a hemispherical magnetic levitation haptic interface device," in *SPIE Int. Symposium on Intelligent Systems and Intelligent Manufacturing, (Proc. SPIE)*, Vol. 3602, Greensburg PA, Sept. 1997.

[7] Brooks, F.P., Ouh-Young, M., Batter, J.J., Jerome, P., "Project GROPE — Haptic Displays for Scientific Visualization," *Computer Graphics (proc. SIGGRAPH 90)*, pp. 177-185, Aug. 1990.

[8] Cohen, J.D., Lin, M.C., Manocha, D., and Ponamgi, M.K., "I-COLLIDE: An Interactive and Exact Collision Detection System for Large-Scale Environments," *Computer Graphics (proc. SIGGRAPH 95)*, pp. 189-196, Aug. 1995.

[9] Clover, C.L., *Control system design for robots used in simulating dynamic force and moment interaction in virtual reality applications*, Ph.D. thesis, Iowa State University, Ames, IA, Apr. 1996.

[10] Colgate, J.E., Grafing, P.E., Stanley, M.C., and Schenkel, G., "Implementation of Stiff Virtual Walls in Force-Reflecting Interfaces," *Proc. IEEE Virtual Reality Annual International Symposium (VRAIS)*, Seattle, WA, pp. 202-208, Sept., 1993.

[11] Craig, J.J., *Introduction to Robotics: Mechanics and Control.* 2nd ed., Addison-Wesley, Reading MA, 1989.

[12] Garcia-Alonso, A., Serrano, N., and Flaquer J., "Solving the Collision Detection Problem," *IEEE Computer Graphics and Applications*, vol. 14, no. 3, pp. 36-43, 1994.

[13] Gottschalk, S., Lin, M.C., Manocha, D., "OBBTree: A Hierarchical Structure for Rapid Interference Detection," *Computer Graphics (proc. SIGGRAPH 96)*, pp. 171-180, Aug. 1996.

[14] Jackins, C., and Tanimoto, S.L., "Oct-Trees and Their Use in Representing Three-Dimensional Objects," *Computer Graphics and Image Processing*, vol. 14, no. 3, pp. 249-270, 1980.

[15] Kaufman, A., Cohen, D., Yagle, R., "Volume Graphics," *IEEE Computer*, 26(7), pp. 51-64, July, 1993.

[16] Logan, I.P., Wills D.P.M., Avis N.J., Mohsen, A.M.M.A., and Sherman, K.P., "Virtual Environment Knee Arthroscopy Training System," *Society for Computer Simulation, Simulation Series*, vol. 28, no. 4, pp. 17-22, 1996.

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

[18] Massie, T.H. and Salisbury, J.K., "The Phantom Haptic Interface: A Device for Probing Virtual Objects," *Proc. of the ASME International Mechanical Engineering Congress and Exhibition*, Chicago, pp. 295-302, 1994.

[19] Mirtich, B. and Canny, J., "Impulse-based Dynamic Simulation." *Proceedings of Workshop on Algorithmic Foundations of Robotics*, Feb. 1994.

[20] OpenGL Architecture Review Board, Woo, M., Neider, J., and Davis, T. *OpenGL Programming Guide*, 2nd, Addison-Wesley, Reading, MA, 1997.

[21] Ruspini, D.C., Kolarov, K., and Khatib, O., "The Haptic Display of Complex Graphical Environments," *Computer Graphics (Proc. SIGGRAPH 97)*, pp. 345-352, Aug. 1997.

[22] Sclaroff, S. and Pentland, A., "Generalized Implicit Functions for Computer Graphics," *Computer Graphics (Proc. SIGGRAPH 96)*, pp. 247-250, July, 1991.

[23] Witkin A. and Welch, W., "Fast Animation and Control of Nonrigid Structures," *Computer Graphics (Proc. SIGGRAPH 90)*, pp. 243-252, Aug. 1990.

[24] Yokokohji, Y., Hollis, R.L., and Kanade, T., "What you can see is what you can feel. Development of a visual/haptic interface to virtual environment," *Proc. IEEE Virtual Reality Annual Int. Symposium (VRAIS)*, pp. 46-53, Mar., 1996.

[25] Zilles, C.B. and Salisbury, J.K., "A Constraint-based God-object Method for Haptics Display," *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Pittsburgh, PA, pp. 146-151, 1995.