# Six Degree-of-Freedom Haptic Display of Polygonal Models

**Arthur Gregory Ajith Mascarenhas Stephen Ehmann**
**Ming Lin and Dinesh Manocha**
Department of Computer Science
University of North Carolina at Chapel Hill
geom@cs.unc.edu
http://www.cs.unc.edu/∼geom/6DHaptics

**Abstract:** We present an algorithm for haptic display of moderately complex polygonal models with a six degree of freedom (DOF) force feedback device. We make use of incremental algorithms for contact determination between convex primitives. The resulting contact information is used for calculating the restoring forces and torques and thereby used to generate a sense of virtual touch. To speed up the computation, our approach exploits a combination of geometric locality, temporal coherence, and predictive methods to compute object-object contacts at kHz rates. The algorithm has been implemented and interfaced with a 6-DOF PHANToM Premium 1.5. We demonstrate its performance on force display of the mechanical interaction between moderately complex geometric structures that can be decomposed into convex primitives.

**CR Categories:** I.3.6 [Computer Graphics]: Methodology and Techniques - Interaction Techniques

**Additional Keywords:** haptics, virtual reality, force-feedback devices, interactive computer graphics

## 1 Introduction

Extending the frontier of visual computing, haptic interfaces, or force feedback devices, have the potential to increase the quality of human-computer interaction by accommodating the sense of touch. They also provide an attractive augmentation to visual display and enhance the level of immersion in a virtual world. They have been effectively used for a number of applications including molecular docking, manipulation of nano-materials, surgical training, virtual prototyping and digital sculpting. Given their potential, a number of research prototypes and commercial devices that can accommodate up to seven degrees of freedom (DOF) [Bur96] have been designed.

Compared with visual and auditory display, haptic rendering has extremely demanding computational requirements. In order to maintain a stable system while displaying smooth and realistic forces and torques, haptic update rates must be as high as 1000 Hz. This involves accurately computing all the contacts between the object attached to the probe and the simulated environment, as well as the restoring forces and torques – all in less than one millisecond.

Some of the commonly used haptic devices include the 3-DOF PHAMToM arm [MS94] and the SARCOS Dexterous Arm [NNHJ98] that compute point-object contacts and provide only force feedback. However, many applications like scientific exploration, virtual prototyping (e.g. assembly planning and maintainability studies), medical simulation and tele-operation need to simulate arbitrary object-object interactions. A 6-DOF haptic device, that provides torque feedback in addition to force display within a large translation and rotational range of motion, is very useful for such applications. It gives a user the much needed dexterity to feel, explore and maneuver around other objects in the virtual environment.

Although a number of commercial and research 6-DOF haptic devices are becoming available, their applications have been limited. This is mainly due to the complexity of accurate calculation of all the contacts and restoring forces that must be computed in less than one millisecond, as outlined by McNeeley et al. [MPT99]. The existing fast haptic rendering algorithms developed for 3-DOF haptic devices primarily deal with single point contact with the virtual models [ST97, JC98, GLGT99, RKK97] and are not directly applicable to accurately compute object-object contacts. Up to date, only *approximate* haptic rendering technique for mostly static environments, such as the one based on point-voxel sampling, has been proposed for haptic display using a 6-DOF haptic device [MPT99]. Current algorithms for exact contact determination and response computation for general polygonal models are unable to meet the real-time requirements of 6-DOF haptic rendering.

### 1.1 Main Results

In this paper, we present a novel algorithm for haptic display of moderately complex polygonal environments using a six degree-of-freedom force feedback device at kHz update rates. We assume that each object can be decomposed into convex primitives. Since the probe's position changes little between successive frames, our algorithm keeps track of the pairs of closest features between convex primitives [LC91]. By exploiting the temporal coherence and spatial locality of closest features, we can take advantage of incremental computation by caching the last closest feature pair and performing a "greedy walk" to the current pair of closest features. We exploit motion coherence from extremely high haptic update rates to predict contact locations and minimize penetration between the probe and the virtual environment. The restoring forces are computed based on a concept similar to "virtual proxy" [RKK97] extended to 6-DOF force feedback devices. The algorithm has been applied to haptic display of mechanical interaction between moderately complex structures composed of tens of convex primitives.

As compared to earlier approaches, our algorithm offers the following advantages:

- **Applicability to Dynamic Environments:** We do not assume the environment is static. Objects in the scene are free to move simultaneously. Our current implementation can only handle a few moving objects at the same time.

- **Accurate Contact Determination:** We do not need to trade off accuracy for performance, while maintaining required force update rates.

- **Smooth Collision Response:** By an extension of the virtual proxy concept, our system generates a natural haptic response that does not result in force discontinuities.

### 1.2 Organization

The rest of the paper is organized as follows. Section 2 briefly surveys related work. Section 3 presents an overview of our approach. Section 4 describes the algorithm for contact determination and penetration depth estimation. Section 5 describes contact force and torque computation. Finally, we describe the implementation in Section 6 and highlight its performance on different environments.

## 2 Previous Work

In this section, we survey previous work related to haptic rendering, contact determination and collision response.

### 2.1 Haptic Display

Several techniques have been proposed for integrating force feedback with a complete real-time virtual environment to enhance the user's ability to perform interaction tasks [CB94, MS94, SBM+95]. Iwata describes a 6-DOF haptic master and the concept of time-critical rendering at a lower update rate of hundreds Hz [Iwa90]. Ruspini et al. [RKK97] presented a haptic interface library "HL" that uses a virtual proxy and a multi-level control system to effectively display forces using 3-DOF haptic devices. Thompson et al. [TJC97] have presented a system for direct haptic rendering of sculptured models.

### 2.2 Contact Determination

The problem of fast collision detection and contact determination has been well studied in computational geometry, robotics and graphics literature.

**Convex Polytopes:** A number of specialized algorithms have been developed for contact determination and distance computation between convex polytopes [GJK88, Bar90, LC91, DHKS93, GHZ99, EL00].

**Hierarchical Approaches:** Some of the commonly used algorithms for general polygonal models are based on hierarchical data structures. These include bounding volume hierarchies where the bounding volume may correspond to a sphere [Hub95], axis-aligned bounding box, oriented bounding box [GLM96], a k-DOP [KHM+98] or a swept sphere volume [LGLM99]. These algorithms can only compute all pairs of overlapping triangles and not the intersection region. Algorithms for separation distance computation based on bounding volume hierarchies have been proposed by [Qui94, JC98, LGLM99].

**Intersection Region and Penetration Depth:** Given two polyhedral models, algorithms for intersection computation and boundary evaluation have been extensively studied in solid modeling. Bouma and Vanecek [BV93] used spatial partitioning approaches to compute the contact region. Ponamgi et al. [PML97] combined hierarchical representations with incremental computation to detect contacts between polygonal models. Snyder [Sny95] used temporal coherence to track penetration depth on spline models.

All these algorithms rely on a surface-based representation of the model. Their performance varies as a function of the size and relative configuration of two models.

**Volumetric Approaches:** Gibson [Gib95], Avila and Sobierajski [AS96] have proposed algorithms for object manipulation including haptic interaction with volumetric objects and physically-realistic modeling of object interactions. More recently, McNeely et al. [MPT99] have proposed a voxel sampling technique for 6-DOF haptic rendering, where point samples from one surface are tested against

the voxel representations of the static environment. This approximation approach achieves constant query time at the expense of accuracy and correctness of haptic force display.

### 2.3 Collision Response

There is considerable work on dynamic simulation and response computation [Bar90, Bar94, MC95]. However, these algorithms do not guarantee real-time performance. Other algorithms propose an artificial coupling between the haptic display and the virtual environment [Cea94] or the notion of a "god object" [ZS95, RKK97].

## 3 Overview

In this section, we give an overview of our approach. We highlight our rendering algorithm for 6-DOF haptic devices. The same methods are applicable to other devices as well.

### 3.1 Preliminaries

In general, the computation for haptic rendering at each time frame involves the following steps:

1. **Collision Detection** - The algorithm first detects if an intersection has occurred between an object (or a probe) held by the user and the virtual environment.

2. **Computing the Contact Manifold** - If an intersection has occurred, then the algorithm needs to compute the intersection points that form the *intersection region* and the *contact normal direction*.

3. **Estimating the Penetration Depth** - A measure or an estimation of penetration depth along the contact normal direction is computed from the intersection region.

4. **Computing Restoring Forces and Torques** - A restoring or contact force is often calculated based on penalty methods that require the penetration depth. Given the force and the contact manifold, restoring torques can be easily computed.

The *contact manifold* refers to the set of all points where the two objects come into contact with each other or may come into contact for predictive methods. For haptic simulation, the stability of force computation is extremely important. An accurate computation of the contact manifold at each frame also helps in smoothing the transition of force display from one frame to the next. This is crucial for the stability of the force feedback system with a human-in-the-loop, especially in a situation when there are multiple contacts between the object attached to the probe and the simulated environment.

None of the current algorithms and systems can perform force display of interaction between general polygonal models in an efficient and accurate manner. Given the time constraint of performing all these computations in less than a millisecond, our approach uses predictive techniques and incremental computation along with spatial and temporal coherence. As part of pre-computation we initially decompose each object into *convex primitives* [BD92], if such a decomposition is not given. For the rest of the paper, we will refer to them as *primitives*. By exploiting the convexity of the objects, we can compute the contacts between a pair of convex primitives in expected *constant time*, with some simple preprocessing. As long as the number of contacts are bounded, our algorithm can guarantee the performance of force update rates.

### 3.2 6-DOF Haptic Rendering with A Virtual Proxy

The computation of our 6-DOF haptic rendering involves three components: (1) the use of virtual proxy [RKK97], (2) contact manifold computation and penetration depth estimation, and (3) haptic response computation.

**Extension of the Virtual Proxy:** A virtual proxy is a representative object that substitutes for the physical object that the probe is attached to in the virtual environment. The motion of the virtual proxy is greedy in nature. It will move as long as it is not obstructed. Once it runs into a surface of some object in the environment, its motion is constrained to the surface location, in such a way that the actual object position will locally minimize the penetration depth. This is accomplished by predicting where the collision may occur between the two objects, based on the proxy position and velocity (also the actual probe position and velocity) from the previous frame and a simple linear interpolation along the travel path. At the beginning of the current frame, the object attached to the probe is constrained to travel no more than a safe threshold distance based on the predicted scheme, so as to minimize the amount of penetration.

**Contact Determination and Penetration Depth Estimation:** The contact determination algorithm initially narrows down pairs of primitives on different objects that are in close proximity using a combination of an N-body "sweep and prune" test and "real-time scheduling". Then, it checks those pairs in close proximity for contacts. For a pair of convex primitives, we use an expected constant time algorithm to track the pair of closest features, thereby computing the contact manifold, the contact normal and the estimated penetration depth. Moreover, a predictive approach is used to minimize penetration computations between the probe and the virtual environment. This is described in Section 5.

**Collision Response:** The algorithm uses penalty methods to compute a force that is proportional to the penetration depth.

It is then applied to the contact manifold in the direction of the contact normal.

# 4 Contact Manifold & Depth Estimation

In this section, we present our algorithm for computing the contact manifold and estimating the penetration depth. Our algorithm uses convex decomposition to subdivide each object into convex primitives. The algorithm for contact determination uses a two phase approach. In the first phase, it narrows down pairs of primitives that are in close proximity. The algorithm computes a tight fitting axis-aligned bounding box for each primitive using incremental methods. It checks these bounding boxes for overlap by projecting them onto the coordinate axes and sorting them using insertion sort [CLMP95]. In the second phase, the algorithm performs exact contact determination tests on all pairs whose bounding boxes overlap.

## 4.1 Contact Determination between Convex Parts

We use a closest-feature tracking algorithm based on Voronoi regions for convex primitives, first proposed by Lin and Canny [LC91]. It is an incremental algorithm that keeps track of closest features between convex primitives from the previous frame. A *feature* corresponds to a vertex, edge or face of the primitive. Using the external Voronoi regions of the convex primitives, the algorithm marches towards the new set of closest features in a greedy manner. The proof of correctness and the analysis of this algorithm is given in [Lin93].

**Expected Constant Time Performance:** The running time of this algorithm is $O(c)$, where $c$ is the number of features it traverses and is typically much smaller than the total number of features on a given polytope. The number of features that is traversed is expected to be constant due to coherence. When the bounding boxes of a pair of convex primitives overlap for the first time, coherence does not exist. In that case we use a directional lookup table [EL00] that has been precomputed for each convex primitive. It consists of nearest vertices to certain samples on a bounding sphere. The directional lookup table provides the means to quickly find out which vertex of a primitive is near a given direction by a simple lookup. The size of this table is determined by the resolution of directions on a unit sphere and is generally set to a constant. Given a constant size table, the table look up time is also constant. Given the centers of two convex primitives and the vector connecting them, each primitive's table is used to look up a vertex in order to initialize the closest feature tracking algorithm. This method can be used to help restore coherence and the closest features can be determined in expected constant time (a few microseconds) [EL00]. The directional lookup table we used takes up less than a kilobyte for each convex primitive.

**Contact Manifold Computation:** The tracking algorithm always returns a pair of closest features for initializing contact manifold computation. There are six different feature combinations possible. If one of the features is a vertex, then the vertex and the closest point to it on the other feature are used. If both the features are edges, the closest points on the edges are used, assuming they are not parallel. If the two edges are parallel, the algorithm computes the projection of one edge to the other. For the edge-face and face-face cases, the algorithm uses a combination of edge-clipping and face-clipping routines.

**Estimating Penetration Depth:** The penetration depth is computed by extending the closest-feature algorithm. It is defined as the smallest distance that one of the primitives has to move so that the two primitives are just touching and not penetrating along the contact normal direction. This can be computed using the pseudo internal Voronoi region of each primitive [PML97]. For a convex polytope, the boundaries of the pseudo internal Voronoi region correspond to lines and planes, as opposed to quadric surfaces for general polyhedra. The pseudo internal Voronoi regions can be used to track and find all features that form the intersection volume. Given the intersection volume, we compute the penetration depth along the direction of the motion.

## 4.2 Real-Time Scheduling

At each frame, the algorithm initially uses the sweep-and-prune technique [CLMP95] to compute pairs of primitives in close proximity. The complexity of the sweep-and-prune algorithm is expected linear time (with a small constant, $t_s$) in terms of the number of primitives that the objects have been decomposed into. Let the total number of primitives be $N$ and the total number of potential contacts be $K$. The total computation time for each update, $T_h$, for 6-DOF haptic rendering is bounded by:

$$T_h = N \times t_s + K \times t_c + M \times t_f$$

where $t_f$ is the time required for computing the restoring force and torque for each of $M$ resulting contact pairs, and $t_c$ is an upper bound on the runtime performance of the closest features tracking between convex primitives. For large environments, it is possible that the algorithm cannot check all possible pairs for exact contacts in less than a millisecond.

As a result, we use a scheduling scheme to prioritize the pairs which will be checked for contact based on their importance. The pairs are assigned a priority based on:

- Pairs of objects that were in contact in the last frame, are given the highest priority.

- The algorithm prioritizes the remaining pairs based on the amount of time since they were last checked for a contact (the greater the time, the higher the priority).

Given these criteria, the algorithm uses a simple greedy strategy. It sorts all the pairs based on increasing priority values into an array of fixed length. The length of the array is some constant derived from the maximum number of contacts the system can handle and still maintain the force update rate. The time of the pairs in this array is then updated to be that of the current frame, and they are checked for contacts.

# 5 Contact Forces and Torques

Given the contact manifold and estimated penetration depth between the probe and the virtual environment, we can compute the contact forces and torques for 6-DOF haptic rendering. In this section, we describe the basic formulation of 6-DOF force display and the use of predictive techniques to avoid penetration as much as possible.

## 5.1 Restoring Forces and Torques

We compute the restoring or contact forces based on the penalty methods. Using Hooke's law, we generate a spring force $F_r$ that is proportional to the penetration depth:

$$F_r = k_s D_p,$$

where $k_s$ is the spring stiffness constant and $D_p$ is the depth of penetration. We use 0.6 N/mm as the value of $k_s$ in our implementation. The computed restoring force vector $\mathbf{F}_r$ is applied to the contact manifolds along the contact normal direction to resolve the penetration, thereby generating a sense of touch.

Restoring torques are generated by

$$\mathbf{T}_r = \sum_i \mathbf{R}_i \times \mathbf{F}_{ri},$$

where $\mathbf{F}_{ri}$ is the contact force vector applied at the point $p_i$ and $\mathbf{R}_i$ is the radius vector from the center of mass to $p_i$.

## 5.2 Predictive Collision Response

The computation of penetration depth is expensive in general. Furthermore, our depth estimation algorithm only computes a local approximation to the penetration depth. In conjunction with the use of virtual proxy, we minimize the frequency of computing the penetration depth by conceptually "growing" the actual surface along its surface normal direction by some small amount $\delta$. Whenever the distance between two objects is less than $\delta$, say $d < \delta$, we declare a collision. Then, we apply a restoring force

$$F_r = k_s(\delta - d).$$

If an actual penetration of $D_p$ occurs, then we modify the contact force using the same principle by setting

$$F_r = k_s(\delta + D_p).$$

This formulation reduces the need for computing penetration depth, which is relatively more expensive than computing the separation distance.

The value $\delta$ is a function of the upper bound on the magnitude of the current velocity $v_c$, which takes into consideration both the linear and angular velocity of the moving objects. $\delta$ is set to be:

$$\delta = v_c \times \Delta t$$

where $\Delta t$ is 1 ms for typical haptic force update.

## 5.3 Force and Torque Interpolation

The displayed force is computed as a function of: penetration depth, contact manifold and contact normal direction. It is possible that the magnitude of contact forces can vary, thereby creating sudden jumps and introduce sharp discontinuities between successive frames. We use a notion of *force and torque interpolation* that adopt the interpolated normals for "force shading", similar to that presented in [RKK97]. We interpolate between two different force normals to achieve smooth force shading effects. In addition, we use a simple linear smoothing scheme to minimize discontinuity in force and torque display between successive frames.

Let $F_0$ be the force displayed at the previous frame and $F_1$ be the force generated during the current frame. Without loss of generality, let us assume that $F_1 > F_0$. Let $F_{max}$ be the maximum amount of force difference allowed between successive updates. We use the following formulation to display the restoring force $F_1$:

if $(F_1 - F_0) > 2F_{max}$
  then $F_1 = F_0 + F_{max}$
  else if $F_1 - F_0 > F_{max}$
    then $F_1 = (F_0 + F_1)/2$
DISPLAY $F_1$

We have considered other higher order smoothing functions. However, this formulation seems to work reasonably well and is simple to compute. Due to the very fast force update rate, a more complex smoothing function may take unnecessarily long to compute and is likely to only result in very minute and subtle differences in the force and torque feedback to the user.

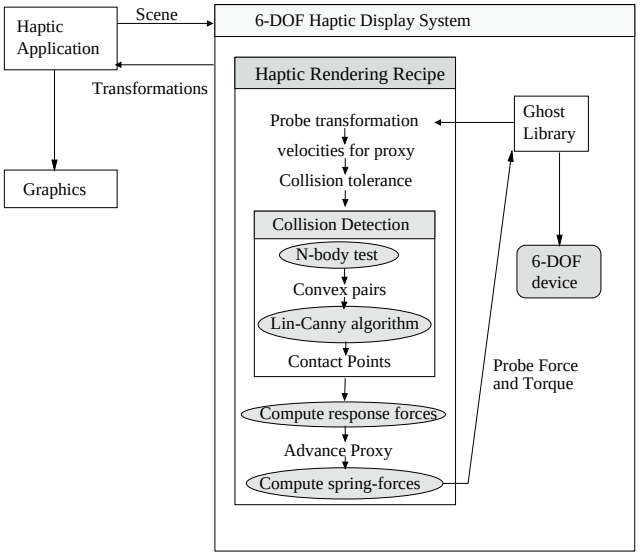


**Figure 1.** Overall architecture of the 6-DOF haptic display system

| System component | Time (ms) |
|---|---|
| Number of updates = 12499 | |
| Worst Time | 0.34300 |
| Average Time | 0.178922 |
| Total Col. Det. tests = 12499 | |
| Average Col. Det. time | 0.071615 |
| Total N-body tests = 7285 | |
| Average N-body time | 0.040022 |
| Total Exact CD tests = 38989 | |
| Average Exact CD time | 0.001291 |
| Total number of collisions = 10870 | |
| Average Response time | 0.092331 |

**Table 1.** Timings for the different system components. The average timing is further broken down into its various components. The collision detection calls normally include an N-body test and typically a few exact collision tests plus the time to determine the closest features and to compute the contact normal for each contact feature pair.

# 6 System Implementation and Performance

In this section, we describe our system and its application to force display of mechanical interaction between moderately complex structures. We used the 6-DOF PHANToM Premium 1.5 device [Che99] designed by SensAble Technologies. It provides intuitive force as well as torque feedback.

## 6.1 Implementation

The system consists of routines for hierarchical transformations, predictive methods, collision detection and contact manifold computation, and computing contact force/torque. It is interfaced with the provided $GHOST^{TM}$ library as well as graphical display routines. The overall architecture of the system is shown in Figure 1.

The main loop of the system consists of three basic parts:

- *Contact Determination*: The probe proxy is typically attached to some object in the scene. The linear and angular velocities for this object are computed from the difference in the transformation between the proxy and the probe. Given the time slice for the current frame and the velocities of the moving objects, a conservative estimate is computed as a tolerance for collisions. Contact determination initially consists of applying an N-body algorithm based on sweep and prune. If the environment has too many potential pairs, it makes use of the scheduling scheme to prioritize the pairs as described in Section 4.2. For each pair in close proximity, it checks them for contact and computes the contact manifold between each pair. For convex objects the contact manifold can be used to compute a contact normal and a pair of contact points to apply the response forces.

- *Collision Response*: For each pair in contact, the algorithm uses the relative velocity at the contact points, the contact normal and the distance between them to estimate the time of impact. If there are multiple contacts during the given time frame, the algorithm uses the one with the smallest time of impact. Each object is then advanced using its current velocities to the time of impact, and the contact forces are applied. Finally it is advanced with its new velocities for the remainder of the time in the frame.

- *Computing Proxy Forces*: The actual force and torque being applied to the probe are computed using a simple spring model based on the transformation from the actual probe to its proxy at the end of the frame. After that a simple smoothing is applied to the force and torque vectors to make sure that the first derivatives of both the magnitude and the direction of these vectors have not changed drastically between frames.

- *Real-Time Constraints:* An important component of a high-fidelity haptic rendering system is that all the computations including contact determination and response computation have to be performed in less than a millisecond. In practice, this is a rather hard constraint and many portions of our system need to be optimized to achieve such performance. Currently, our system only works well on geometric models that can be decomposed into a few convex primitives ($10 - 30$

convex primitives). The algorithm for collision detection between a pair of convex primitives takes just a few microseconds.

Average running times for each phase of the force/torque display during a haptic session are shown in Table 1.

## 6.2 System Demonstration

We have applied this system for force and torque display to several applications: gear turning, inserting a peg in a hole, and interaction with multiple moving objects.

• **Mechanical Interaction Between Gears:**

The gears are modeled and positioned in such a manner that the user can turn one gear with the other in either direction. The teeth are not so tightly interlocking that there is always a collision. In the demonstration shown, the gears contain a few hundred polygons and about 20 convex primitives, as shown in Color Plate I (Left).

In order to aid the user in turning one gear with the other, constraints are applied to the gears so their position cannot be altered, and their rotation is constrained to be about an axis. In order to generate a correct response these constraints are applied to the velocities at the beginning of the haptic frame, and to the new transformations computed for the objects towards the end. As a result the user can feel the gears rotating against each other just as they would in a real mechanical part, as shown in Color Plate I (Right).

The user can attach the haptic probe to either gear in two different modes. Clicking the button towards the rim of the gear is like inserting a rod into a bicycle wheel. The probe's position remains at a fixed radius from the center of the gear and its orientation is fixed so as to rotate with the gear at that radius. If the user clicks around the center of a gear, the probe effectively becomes the gear. Its position is fixed, and it is only allowed to rotate about one axis. In this mode, the user turns the gears and feels their interactions entirely through the torque of the probe handle. We encourage the reader to view the system demonstration on our website:

http://www.cs.unc.edu/~geom/6DHaptics.

• **Inserting a Peg in a Hole:**

In this scenario, the user attaches the probe to a rectangular peg and attempts to insert it into a rectangular hole. Often one or two pairs of the parallel faces are the pair(s) of the closest features and may be in contact. If any type of sampling technique is used, the number of contact points would be very high, since nearly all faces of the peg are in close proximity with the walls of the hole. Collision detection and contact determination for such a seemingly simple scene are actually rather difficult, due to its contact configuration and geometric robustness problem. A sequence of snapshots are given in Color Plate II to demonstrate a successful attempt of a user inserting the peg into the hole with 6-DOF haptic display.

• **A Dynamic Scene of Multiple Moving Objects:**

In this scenario, all objects are moving simultaneously under the influence of gravity and impact due to collision with other objects. The user can pick up any of the objects with the probe and move it to hit other objects or feel other objects hitting it.

Sample snapshots are shown in Color Plate III. In this particular setup, there are four cubes, four spheres (320 faces each), four ellipsoids (320 faces each) and a stick-like block. There are two types of simulated force. There is continuous force/torque such as gravity. There is also impulsive force/torque due to impact between the user controlled object and other moving objects. The motion of all moving objects is simulated using impulse-based rigid body dynamics. The continuous force can be felt quite well but the impulsive ones currently feel like small blips. This is exactly what we expect since the impulsive contact duration is very short. We are considering the possibility of force expansion over time or force amplification to exaggerate the feel of impulsive force/torque.

## 6.3 Discussion

Force display of mechanical interaction is useful in virtual assembly, maintenance studies and other electronic prototyping applications. In cases where the user has to interact with the virtual environment through a sense of touch (e.g. a mechanic trying to remove a virtual part from a virtual engine), haptic display appears to be the only means of human-computer interaction. In many other cases (e.g. molecular graphics [BOYBK90]), force display can provide additional means to visualize complex systems or environments.

As overall scenes become more complex, the type of contact scenarios do *not* necessarily become more complex, since the contact configuration in most cases is only local to the region of impact. In fact, in our gear-turning demonstration, the polygon count is much higher than the *visually simple* peg-in-the-hole scenario. However, the collision detection and contact determination problem becomes substantially harder for peg-in-the-hole insertion, since nearly the entire peg is in contact with the hole. Although there are already many contact pairs between the interlocking gears, there are significantly many more contacts (in theory infinitely many point contacts) in the peg-in-the-hole case. Haptic display becomes much more difficult to control due to multiple contact forces generated in opposing directions. This is an extremely challenging scenario. Our approach can provide a more accurate and smoother response than an approximate method. However, if the user exerts too much force which causes a large amount of penetration, the con-

trol loop can become unstable or the device can shut down due to the force limit being exceeded. Furthermore, if we incorporate a more complex and accurate dynamics model for simulating sliding/rolling friction, it would become even more difficult to maintain the required force update rate on hard surfaces. This is a research area deserving serious investigation.

# 7 Conclusions and Future Work

In this paper, we have presented a haptic rendering algorithm using 6-DOF force feedback devices. It makes use of a combination of incremental techniques, spatial and temporal coherence and predictive methods to compute all the contacts accurately and force response in less than a millisecond. It has been used for force display of mechanical interaction. In terms of complexity, our current system works well on geometric models that can be decomposed into few tens of convex pieces.

There are many directions for future work. We plan to use 6-DOF haptic rendering for display of more complex geometric models, as well as large vector field datasets. In terms of haptic display of complex polygonal models, a major issue is to accurately compute all the contacts and the penetration depth between general polygonal models in less than 1 millisecond. Algorithms based on hierarchical approaches, e.g. OBBTree [GLM96], cannot guarantee to compute all the contacts in less than a millisecond for complex models. Extensions based on the approach presented in this paper hold some promise. In addition, we would also like to interface our system with scientific visualization systems for better understanding and analysis of complex datasets. We have developed a generic system framework for haptic visualization of force fields (represented as volumetric data) using a 6-DOF force feedback device and have successfully applied it to some complex volumetric datasets [MLM00]. Finally, we would like to perform some user studies on the benefits of multi-modal display systems using 6-DOF force display over traditional visualization techniques.

**Acknowledgments:** We would like to thank the anonymous reviewers for their helpful comments. This research is supported in part by ARO DAAG55-98-1-0322, DOE ASCI Grant, NSF NSG-9876914, NSF DMI-9900157, NSF IIS-9821067, ONR Young Investigator Award and Intel.

# References

[AS96] R. S. Avila and L. M. Sobierajski. A haptic interaction method for volume visualization. *Proceedings of Visualization'96*, pages 197–204, 1996.

[Bar90] D. Baraff. Curved surfaces and coherence for non-penetrating rigid body simulation. *ACM Computer Graphics*, 24(4):19–28, 1990.

[Bar94] David Baraff. Fast contact force computation for nonpenetrating rigid bodies. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94*, pages 23–34. ACM SIGGRAPH, 1994.

[BD92] C. Bajaj and T. Dey. Convex decomposition of polyhedra and robustness. *SIAM J. of Comput.*, (No. 2):339–364, 1992.

[BOYBK90] Frederick P. Brooks, Jr., Ming Ouh-Young, James J. Batter, and P. Jerome Kilpatrick. Project GROPE — Haptic displays for scientific visualization. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 177–185, August 1990.

[BV93] W. Bouma and G. Vanecek. Modeling contacts in a physically based simulation. *Second Symposium on Solid Modeling and Applications*, pages 409–419, 1993.

[Bur96] G. Burdea. *Force and Touch Feedback for Virtual Reality*. John Wiley and Sons, 1996.

[Che99] E. Chen. Six degree-of-freedom haptic system for desktop virtual prototyping applications. In *Proceedings of the First International Workshop on Virtual Reality and Prototyping*, pages 97–106, 1999.

[CLMP95] J. Cohen, M. Lin, D. Manocha, and M. Ponamgi. I-collide: An interactive and exact collision detection system for large-scale environments. In *Proc. of ACM Interactive 3D Graphics Conference*, pages 189–196, 1995.

[CB94] J. E. Colgate and J. M. Brown. Factors affecting the z-width of a haptic display. *IEEE Conference on Robotics and Automation*, pages 3205–3210, 1994.

[Cea94] J. E. Colgate, et al. Issues in the haptic display of tool use. *Proceedings of the ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 140–144, 1994.

[DHKS93] D. Dobkin, J. Hershberger, D. Kirkpatrick, and S. Suri. Computing the intersection-depth of polyhedra. *Algorithmica*, 9:518–533, 1993.

[EL00] S. A. Ehmann and M. C. Lin. Accelerated proximity queries between convex polyhedra by multi-level Voronoi marching. Technical report, Department of Computer Science, University of North Carolina, 2000.

[Gib95] S. Gibson. Beyond volume rendering: Visualization, haptic exploration, and physical modeling of element-based objects. In *Proc. Eurographics workshop on Visualization in Scientific Computing*, pages 10–24, 1995.

[GJK88] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between objects in three-dimensional space. *IEEE J. Robotics and Automation*, vol RA-4:193–203, 1988.

[GLM96] S. Gottschalk, M. Lin, and D. Manocha. Obb-tree: A hierarchical structure for rapid interference detection. In *Proc. of ACM Siggraph'96*, pages 171–180, 1996.

[GLGT99] A. Gregory, M. Lin, S. Gottschalk, and R. Taylor. H-collide: A framework for fast and accurate collision detection for haptic interaction. In *Proceedings of Virtual Reality Conference 1999*, 1999.

[GHZ99] L. Guibas, D. Hsu, and L. Zhang. *H-Walk*: Hierarchical distance computation for moving convex bodies. *Proc. of ACM Symposium on Computational Geometry*, 1999.

[Hub95] P. M. Hubbard. Collision detection for interactive graphics applications. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):218–230, September 1995.

[Iwa90] Hiroo Iwata. Artificial reality with force-feedback: development of desktop virtual space with compact master manipulator. In *Proc. of SIGGRAPH*, pages 165–170, August 1990.

[JC98] D. Johnson and E. Cohen. A framework for efficient minimum distance computation. *IEEE Conference on Robotics and Automation*, pages 3678–3683, 1998.

[KHM+98] J. Klosowski, M. Held, J.S.B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Trans. on Visualization and Computer Graphics*, 4(1):21–37, 1998.

[LC91] Ming C. Lin and John F. Canny. Efficient algorithms for incremental distance computation. In *IEEE Conference on Robotics and Automation*, pages 1008–1014, 1991.

[Lin93] Ming C. Lin. *Efficient Collision Detection for Animation and Robotics*. PhD thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley, December 1993.

[LGLM99] E. Larsen, S. Gottschalk, M. Lin and D. Manocha. *Fast Proximity Queries with Swept Sphere Volumes*. Technical Report TR99-018, Department of Computer Science, University of North Carolina at Chapel Hill, 1999.

[MC95] B. Mirtich and J. Canny. Impulse-based simulation of rigid bodies. In *Proc. of ACM Interactive 3D Graphics*, Monterey, CA, 1995.

[MLM00] A. Mascarenhas, M. C. Lin and D. Manocha. Six Degree-of-Freedom Haptic Visualization of Force Fields Technical Report, Computer Science Department, University of North Carolina at Chapel Hill, 2000.

[MPT99] W. McNeely, K. Puterbaugh, and J. Troy. Six degree-of-freedom haptic rendering using voxel sampling. *Proc. of ACM SIGGRAPH*, pages 401–408, 1999.

[MS94] T. M. Massie and J. K. Salisbury. The phantom haptic interface: A device for probing virtual objects. *Proc. of ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 1:295–301, 1994.

[NNHJ98] A. Nahvi, D. Nelson, J. Hollerbach, and D. Johnson. Haptic manipulation of virtual mechanisms from mechanical cad designs. In *Proc. of IEEE Conference on Robotics and Automation*, pages 375–380, 1998.

[PML97] M. Ponamgi, D. Manocha, and M. Lin. Incremental algorithms for collision detection between polygonal models. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):51–67, 1997.

[Qui94] S. Quinlan. Efficient distance computation between non-convex objects. In *Proceedings of International Conference on Robotics and Automation*, pages 3324–3329, 1994.

[RKK97] D.C. Ruspini, K. Kolarov, and O. Khatib. The haptic display of complex graphical environments. *Proc. of ACM SIGGRAPH*, pages 345–352, 1997.

[SBM+95] K. Salisbury, D. Brock, T. Massie, N Swarup, and C. Zilles. Haptic rendering: Programming touch interaction with virtual objects. *Proc. of 1995 ACM Symposium on Interactive 3D Graphics*, pages 123–130, 1995.

[Sny95] J. Snyder. An interactive tool for placing curved surfaces without interpenetration. In *Proceedings of ACM Siggraph*, pages 209–218, 1995.

[ST97] SensAble Technologies Inc. $GHOST^{TM}$: Sofware developer's toolkit. *Programmer's Guide*, 1997.

[TJC97] T.V. Thompson, D. Johnson, and E. Cohen. Direct haptic rendering of sculptured models. *Proc. of ACM Interactive 3D Graphics*, pages 167–176, 1997.

[ZS95] C. Zilles and K. Salisbury. A constraint-based god object method for haptics display. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robotics and Systems*, 1995.