6-DOF Haptic Rendering Using Spatialized Normal Cone Search

David E. Johnson, Peter Willemsen, and Elaine Cohen

***Abstract*— This article describes a haptic rendering algorithm for arbitrary polygonal models using a six degree-of-freedom haptic interface. The algorithm supports activities such as virtual prototyping of complex polygonal models and adding haptic interaction to virtual environments. The underlying collision system computes local extrema in distance between the model controlled by the haptic device and the rest of the scene. The haptic rendering computes forces and torques on the moving model based on these local extrema. The system is demonstrated on models with tens of thousands of triangles and developed in an accessibility application for finding collision-free paths.**

***Index Terms*— H.5.2.g Haptic I/O, I.3.7.g Virtual reality, J.6.a Computer-aided design**

I. Introduction

A force-feedback, or haptic, interface engages a user's sense of touch while exploring virtual models and environments. The forces displayed by the haptic device are computed by a process known as haptic rendering. Haptic rendering must not only determine contact between the models in the environment but also the degree to which they penetrate, so some distance measure is needed.

In this paper we present a system for haptic rendering of high-resolution triangulated models (Figure 1) appropriate for activities such as virtual prototyping and adding haptic cues to virtual environments. Virtual prototyping systems attempt to replace the evaluative aspects of physical prototype models with virtual models in a computer. These types of systems have utility in mechanical design and architecture, where physical models can be costly to produce and limit the ability of a designer to quickly test modifications.

A virtual prototyping environment should support activities such as accessibility, assembly, and placement of models. These activities are difficult to perform using purely computational means. The size of the virtual models overwhelm current algorithmic techniques. Haptic interfaces allow the sense of touch to guide the placement of the models in the scene. This type of interaction is very natural and prior haptic virtual prototyping systems have demonstrated the usefulness of including the sense of touch in complex placement tasks. Our algorithm allows virtual prototyping of standard polygonal models with little preprocessing and permits modifying the scene to quickly test changes to a design.

The authors are with the School of Computing, University of Utah.

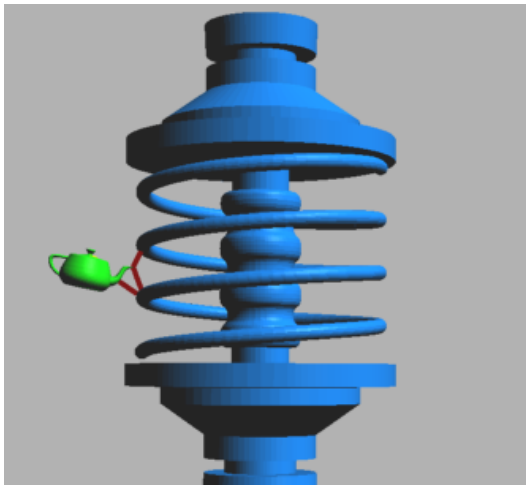

Fig. 1. Our system generates forces between high-resolution triangulated models.

Our system robustly searches for distance extrema using hierarchical normal cones and uses sets of the resulting extrema to provide forces and torques to the haptic interface. The forces derived from these extrema can be used to prevent interpenetration of models, as is desired during accessibility testing of mechanical systems. Alternatively, these extrema can be used to estimate the maximum interpenetration of two models during contact, an approach which allows general haptic exploration.

The normal cone approach differs from prior work in distance computation in that it searches for extrema of a minimum distance formulation in the space of normals, rather than in Euclidean space. Furthermore, it works directly on the triangular model instead of treating the model as a collection of primitives. Finally, the simplicity of the approach allows it to scale to models with tens of thousands of triangles and many local extrema.

Some distinguishing characteristics of our approach

are as follows:

- polygonal models of arbitrary shape can be used in the virtual environment.
- elements of the scene can be moved or added and deleted without requiring substantial preprocessing.
- environments with large number of triangles can be used, increasing the accuracy of simulated model interactions.

The paper is divided into several sections. First, the hierarchical normal cone approach for distance computations is presented. Then, this approach is applied to haptic rendering of triangulated models. We then add an acceleration technique using local tracking, which provides significant speedups. These approaches are demonstrated on a variety of models and on an accessibility application using a haptic interface. Finally, the normal cone approach is adapted to penetration depth computations, demonstrating how our system can handle general haptic rendering queries in complex environments.

II. Related Work

Advancements in haptic rendering and geometric computations have been tightly linked. The following sections review relevant work in distance computations and existing work in haptic rendering.

A. Distance Computations

Almost all the literature on minimum distance, especially for polygonal models, treats the problem primarily as a Euclidean one. Approaches typically partition the model with hierarchical spatial bounding volumes, using primitives such as spheres [1], convex polytopes [2], oriented swept sphere volumes [3], and convex surface patches [4]. Nodes of the hierarchy are pruned by comparing lower bound distances between nodes to upper bounds on the global distance. This approach returns the global minimum distance between models.

A different approach is found in work on sculptured surfaces, such as B-splines [5], [6]. These approaches develop techniques for tracking the locally closest or extremal points between two surfaces after initialization by a global search [7]. The set of equations that describe these local distances is based on collinearity of normals rather than Euclidean distance.

In [8], the collinearity approach is adapted to polygonal models through the spatialized normal cone hierarchy data structure. This technique efficiently finds all local minima between polygonal models.

B. Haptic Rendering

Haptic rendering algorithms were first developed to support three degree-of-freedom (DOF) haptic interfaces. These algorithms and devices support a moving point touching a computer model. Researchers have developed algorithms for haptic point contact with polygonal [9], [10], [11], [12], [13], implicit [14], and NURBS [15] models. Thompson [16] has developed a complete system for three DOF haptic interaction with trimmed NURBS models.

Other efforts have focused on developing techniques to haptically render the interactions between two models. The resulting forces include torques as well as translation forces, and a six DOF haptic device is needed to accurately reflect the results back to a user.

In the polygonal model domain, the first efforts at six DOF haptic rendering were for small convex shapes [17]. More recently, research has looked at collections of convex bodies [18], as well as incremental methods for computing the penetration depth [19]. Most recently, the convex decomposition approach has been extended with perceptual level of detailing to accelerate haptic rendering for very large models [20].

Model-model haptic rendering for general NURBS models is developed in [5]. This system uses a three pass approach: initial distance monitoring using polygonal approximations, local closest point initialization using Newton's method on an extremal distance formulation, and stable maintenance of the penetration depth distance with a velocity relation between parametric space and Euclidean movement. Another method based on control theory maintains the extremal distance even with imprecise starting values [6].

Instead of using penetration-based methods, researchers at Boeing create a voxel-based scene that allows a point-sampled model to interact with the voxels [21]. The advantage is that the computation time can be tightly bound by the number of voxels and the number of points in the free-moving model. They also created a voxel boundary around the models in the scene to prevent interpenetration of models, which would invalidate the correctness of the virtual prototyping.

This article expands on the normal cone approach in [8], by more fully describing the pruning and leaf tests used, and then provides a cohesive view of the normal cone approach's application to haptics in [22] and [23]. In addition, this article presents an extension of the approach to handle haptic rendering of model-model penetration.

III. Approach

Prior work in haptic rendering of point-model contact found local minima in distance by searching for a global distance minimum and then constraining movement from the solution at the previous time step to the new global solution. Instead, our approach is to search directly for local distance minima based on techniques commonly applied to smooth models, such as splines.

For continuous, parametric models, the distance between surfaces $\mathbf{f}(\mathbf{u},\mathbf{v})$ and $\mathbf{g}(\mathbf{s},\mathbf{t})$ is

$$D(u,v,s,t) = ||\mathbf{f}(u,v) - \mathbf{g}(s,t)|| \tag{1}$$

A global minimum can be found generically by solving for all local extrema and returning the smallest. Extrema are found at simultaneous zeros of the partials of Eq. 1, as in

$$\begin{bmatrix} (\mathbf{f}(u,v) - \mathbf{g}(s,t)) \cdot \mathbf{f}_u \\ (\mathbf{f}(u,v) - \mathbf{g}(s,t)) \cdot \mathbf{f}_v \\ (\mathbf{f}(u,v) - \mathbf{g}(s,t)) \cdot \mathbf{g}_s \\ (\mathbf{f}(u,v) - \mathbf{g}(s,t)) \cdot \mathbf{g}_t \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{2}$$

Equation 2 shows that the location of distance extrema depends on the collinearity of normals on each surface to the vector connecting the solution points. We develop a technique called spatialized normal cone hierarchies (SNCH) to search for such distance extrema on polygonal surfaces rather than parametric models. The next section develops SNCH search for local distance minima between two polygonal models and following sections extend it to haptic rendering of model-model contact.

IV. Spatialized Normal Cone Hierarchies

While other collision and distance techniques use hierarchies of conservative spatial bounds to prune away portions of a model, we seek instead to conservatively bound the range of normals at each node in a hierarchy. A cone, defined by an axis vector and a spread angle, provides such a bound.

A. Building the Hierarchy

Our system uses a vertex-edge-face data structure with neighbor information for vertices and edges on top of the triangulated model, rather than an unstructured triangle cloud. Most unstructured models can be collapsed into our mesh format by searching for common vertices and edges. For a given model A, a hierarchy is constructed using the binary spatial splitting technique described in [3]. The leaf nodes point to the original triangles of the model. If normals are not provided they are computed for the triangle faces and estimated for the edges and vertices.

At each node Φ_A in the hierarchy, the face, edge, and vertex normals from contained triangles are averaged to compute a cone axis vector $\vec{C}_{\Phi_A}$. Once the axis is computed, the half spread angle ϕ_{Φ_A} is just the maximum deviation of a contained normal from the axis vector. This cone data is stored in each node.

Each node also stores a bounding sphere with center S_{Φ_A} and radius ρ_{Φ_A} that spatially bounds the contained geometry. We use a simple averaging and maximum deviation scheme to create the sphere. This bounding sphere is used in the search for extrema as described in the next section.

B. Searching the Hierarchy

For a minimum distance search between two models A and B there will be two SNCH structures, one for each model. The top nodes of each structure are connected as an *active pair*. Each active pair undergoes a series of tests that determine whether or not a local minimum for the full models can potentially exist for the contained geometries. An active pair that passes the tests forms four new active pairs connecting the two sets of children nodes, one from each hierarchy, and these new active pairs are recursively tested.

Recalling the system of equations (Eq. 2) that define a distance minimum, the pruning test for an active pair first checks if if there exists a vector in the cone for A collinear with a vector in the cone for B by comparing the angle between cone axes and their spreads. If

$$\pi - \arccos\left(\vec{C}_{\Phi_A} \cdot \vec{C}_{\Phi_B}\right) > \phi_{\Phi_A} + \phi_{\Phi_B} \tag{3}$$

is true, then the active pair is rejected, since no part of one cone is collinear with the other.

If this first test is passed, then a more comprehensive test is needed. Eq. 2 does not specify just that the surface normals must be collinear, but also that the line connecting the closest pair of points is collinear with each surface normal. Because the closest pair of points between each node has not yet been determined (and will not be until the leaf nodes are reached), the spheres that bound the nodes' geometries are used to conservatively bound the range of possible solution lines between the two nodes.

The range of possible solution lines forms a double cone, the *solution line cone*, between the two spheres, with a central axis connecting the spheres' centers and a half spread angle $\sigma_{\Phi_A\Phi_B}$ equal to

$$\sigma_{\Phi_A\Phi_B} = \arcsin\left(\frac{\rho_{\Phi_A} + \rho_{\Phi_B}}{||S_{\Phi_B} - S_{\Phi_A}||}\right). \tag{4}$$

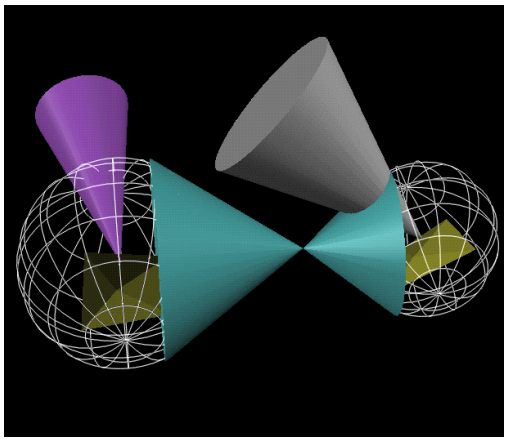

Fig. 2. The normal cones must point toward each other and along the dual solution line cone for a local distance minimum to exist.

The spread angle is mapped to a 0-180 degree range by subtracting the angle from π if the arcsine is negative.

Given the solution line cone, an active pair is tested for whether its contained geometry could satisfy Eq. 2 (Figure 2) by seeing if

$$\begin{aligned}
&\arccos\left[(S_{\Phi_A} - S_{\Phi_B}) \cdot \vec{C}_{\Phi_A}\right] - \sigma_{\Phi_A\Phi_B} - \phi_{\Phi_A} \leq \pi \\
&\arccos\left[(S_{\Phi_A} - S_{\Phi_B}) \cdot \vec{C}_{\Phi_A}\right] + \sigma_{\Phi_A\Phi_B} + \phi_{\Phi_A} \geq \pi \\
&\arccos\left[(S_{\Phi_A} - S_{\Phi_B}) \cdot \vec{C}_{\Phi_B}\right] - \sigma_{\Phi_A\Phi_B} - \phi_{\Phi_B} \leq 0 \\
&\arccos\left[(S_{\Phi_A} - S_{\Phi_B}) \cdot \vec{C}_{\Phi_B}\right] + \sigma_{\Phi_A\Phi_B} + \phi_{\Phi_B} \geq 0.
\end{aligned} \tag{5}$$

Active pairs that do not satisfy all these predicates are pruned. Pairs that pass are subdivided and recursively tested.

C. Leaf Tests

If both nodes are leaves, then an exact test is done on the leaf triangles T_A and T_B. A local minimum, if it exists, must be between the closest points between T_A and T_B. So the first step of the exact leaf test is to find those closest points. The features the closest points are on, either a triangle face, edge, or vertex, are also recorded.

The closest points between leaf triangles form a potential solution vector. The normalized solution vector $\vec{L}_{T_AT_B}$ is tested for collinearity with a vector in the normal range of each triangle's closest feature, to check for compliance with the constraints of Eq. 2.

1) Face Test: If the closest feature on model A is a triangle face, then the associated normal range is just the triangle normal $\vec{N}_{T_A}$. The solution vector $\vec{L}_{T_AT_B}$ must satisfy the collinearity condition for the face

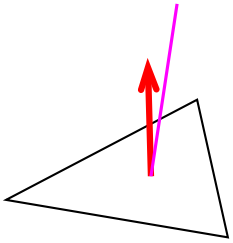

Fig. 3. If the solution line falls on a triangle face it is compared with the face normal for collinearity.

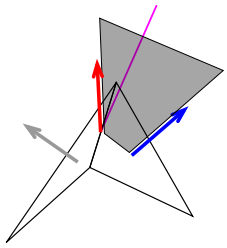

Fig. 4. Edges encompass a range of normals between neighbor faces. Only half the range is tested for each triangle to avoid redundant solutions.

$$\vec{L}_{T_AT_B} \cdot N_{T_A} = 1. \tag{6}$$

2) Edge Test: For a solution vector emanating from an edge, the test is slightly more complex. An edge has a range of vectors between the normals of the two faces that share the edge. However, each triangle only tests from its face normal $\vec{N}_{T_A}$ to the average edge normal $\vec{E}_{T_A}$ in order to avoid redundant solutions with the other triangle that shares the edge. The edge test checks if

$$\left(\vec{L}_{T_AT_B} \times \vec{N}_{T_A}\right) \cdot \left(\vec{E}_{T_A} \times \vec{L}_{T_AT_B}\right) \geq 0. \tag{7}$$

Solution line vectors that meet the range condition are accepted for that edge.

3) Vertex Test: The spread of normals at a vertex is a complicated shape dependent on the number and orientations of the triangles that share the vertex. If the

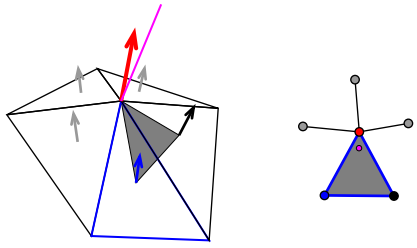

Fig. 5. The vertex test maps each triangle's portion of the vertex normal spread to a triangle on the Gauss sphere, where the solution line can be tested for inclusion.

full vertex normal spread were tested, then each triangle sharing that face could contribute a redundant solution. We adopt the convention of only using the normal range from the vertex average normal $\vec{V}_{T_A}$ to the triangle face normal $\vec{N}_{T_A}$ to the counterclockwise neighbor triangle face normal $\vec{N}_{\curvearrowleft T_A}$.

The test to see if $\vec{L}_{T_A T_B}$ lies within the Gauss map triangle for a vertex first checks for degeneracy in any of the boundary arcs. If any are degenerate, then the test reduces to comparing the solution vector along the remaining arc. If the triangle is non-degenerate, then the test sees if $\vec{L}_{T_A T_B}$ is on the same side of all the consistently oriented edges of the Gauss map triangle, as in

$$\begin{aligned} \vec{L}_{T_A T_B} \cdot \left(\vec{N}_{T_A} \times \vec{V}_{T_A} \right) > 0 \\ \vec{L}_{T_A T_B} \cdot \left(\vec{V}_{T_A} \times \vec{N}_{\curvearrowleft T_A} \right) > 0 \\ \vec{L}_{T_A T_B} \cdot \left(\vec{N}_{\curvearrowleft T_A} \times \vec{N}_{T_A} \right) > 0. \end{aligned} \tag{8}$$

D. Discussion

The closest pairs of points for all leaf active pairs that pass the exact leaf tests are returned as local minima. Local maxima can be found in a similar manner, although we will delay that discussion until a later section. These local minima provide greater information about the relationship of two models than just a global minimum, and we will use this additional information to develop 6-DOF haptic rendering for polygonal models.

V. Haptic Rendering Using SNCH Search

Rather than finding forces that move models apart once they have collided, our haptic rendering algorithm prevents collisions by applying repulsive forces as models approach each other. This technique is appropriate for representing interactions between models since allowing models to penetrate each other violates real-world constraints. Our test system uses a 6-DOF haptic interface as the means of moving a model in the scene and for reflecting the forces of model-model collision back to the user.

A. Local Minimum Distances

Sets of local distance minima are used because while a global minimum can be rapidly computed between polygonal models it would only generate a single penalty force at each time step. This force could rapidly change direction, creating haptic instabilities. One could easily imagine modifying a distance computation to return all pairs that are within a certain distance, rather than just the global minimum. However, this could potentially create very large numbers of penalty forces, which would swamp the haptic computation

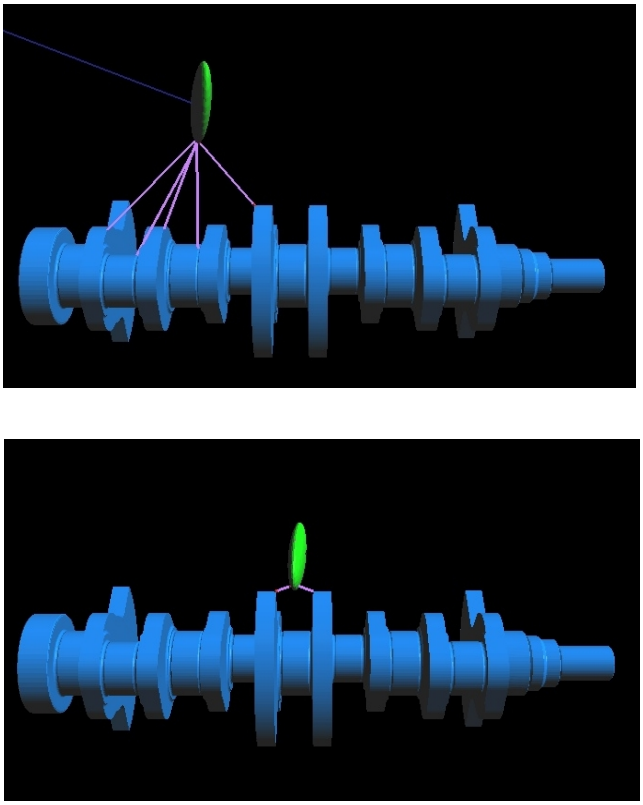

Fig. 6. A cutoff distance focuses the computation near potential contacts.

We argue that an appropriate solution is to compute the local minimum distances (LMD) between models. Imagine two models that have just collided. This collision can be represented at a single point on each surface (even for manifold contacts, a single point encapsulates that area of contact). If the models move apart, this pair of points tracks the local minimum distance and represents the potential future contact between entire sections of these two models. Additional pairs of contact points for those sections are redundant predictors of future contacts for those regions, thus the local minimum distance pairs are adequate. This formulation keeps a manageable number of forces while maintaining coverage of potential contacts.

B. Cutoff Distance

The SNCH search returns all LMDs between two models. Much of that computation is wasted for haptic rendering, where we are only interested in potential

contacts in the immediate vicinity of the moving model. We introduce a cutoff distance to the SNCH search. All active pairs with bounding spheres further apart than the cutoff distance are pruned without further checking (Figure 6). The cutoff distance is dynamically adjustable, allowing for user control.

C. Computing Forces and Torques

At each time step in the haptic rendering loop, our algorithm computes all the LMDs that are closer than the cutoff distance between the model that is controlled by the haptic interface and the rest of the models in the scene. Each LMD is considered a virtual spring with a rest length equal to the cutoff distance. Each spring is attached to the models by the pairs of points that form the LMD.

The center of mass and the first-order moments are approximated by the geometric extent of a PQP generated, oriented swept sphere bounding box [3] surrounding and approximating the shape of the model. More precise values can be used easily when available.

The repulsive forces between models begin at zero at the cutoff distance, so LMDs that are created and destroyed as sections of the two models approach the cutoff distance only modify the total force and torque a small amount. Furthermore, since we are not attempting to render the forces of hard contact, only guiding the placement of models, the springs can be fairly soft, smoothing the haptic rendering.

D. Acceleration with Local Search

We speed updates of LMDs by using a local gradient search while waiting for new global solutions. The haptic rendering system first computes all LMDs within the cutoff distance using the global SNCH search. These LMDs are fed to a local update thread, which performs local gradient descent on the LMDs given new positions of the models. The updated LMDs are used to compute forces and torques repelling the models. The local update works as fast as it can on the LMDs it knows about. Concurrently, the global search computes new LMDs. When it finishes a time step, it notifies the local search that new LMDs are available. The local search then updates these new LMDs to the current model positions and continues local updates.

A pair of points, one on each model, forms each LMD. After a model moves, the local search algorithm looks at the neighborhood around each LMD point and computes the distances between all the triangles in one neighborhood and all the triangles in the other models neighborhood. If any of these triangle pairs are closer than the current LMD, then the search continues with those triangles' neighborhoods until the minimum distance converges. The points that form this new minimum distance are the updated LMD.

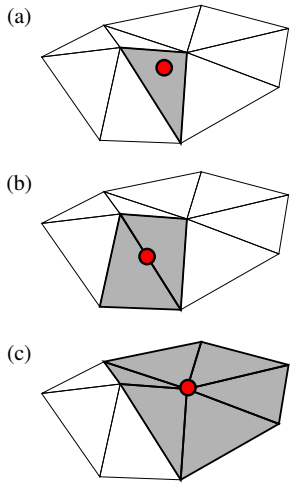

Fig. 7. The local crawl neighborhood depends on if the closest point is on a face, edge, or vertex.

If the last LMD point was on the face of a triangle, then the local neighborhood is just that triangle again (Figure 7). If the last point was on an edge, only the two triangles that share that edge are part of the local neighborhood. When the last point was at a vertex, all triangles that share that vertex are searched for a new LMD.

E. Search Efficiency

The number of neighborhoods that must be checked varies with the model resolution and the movement of the models. For models with low aspect ratio side lengths, the number of triangles searched on one model grows roughly as the $\sqrt{n}$, where n is the number of triangles. In addition, for haptic rendering running near 1000 Hz, temporal coherence is very high and the number of triangles crossed is small.

The global search efficiency is dependent on the number of LMDs as well as the complexity of the two models. In the best case, the two models are of balanced complexity and there is a single LMD, and the global search takes $\log n$ time. In a more typical scenario, there are multiple LMDs and the pruning cannot always remove nodes that will not eventually yield minima, increasing search time. In cases of manifold minima solutions, such as two parallel planes, the search takes $n \log n$ time. However, manifold solutions are rarely encountered with human-guided model interactions. Additionally, the global search is improved when the cutoff

distance is small, so only close portions of the models are searched for minima. This reduces the apparent complexity of the model and the number of LMDs the local update must process.

VI. System Architecture

Our virtual prototyping system is based on a Sensable 6-DOF PHANTOM haptic interface. The computations run on a dual processor Pentium 4 2.4 GHz Linux computer with a gigabyte of RAM and a GeForce 4 Ti 4400 graphics card.

This type of application would be difficult to write as a single thread of computation. The application uses three threads: a global search thread, a local update thread, and a graphics thread. This architecture allows us to restrict the computational load of the graphics and global threads, and let the local update run as fast as possible. On a two-processor system, this translates into the local update getting one processor to itself and the other two threads sharing the other processor.

VII. Examples and Timings

The local search algorithm computes updated forces and torques at kilohertz rates. When model complexity grows, the global search tends to slow down, but the local update speed is mostly dependent on the number of LMDs, not the complexity of the model. The global search is still the limiting factor in environment complexity. We instrumented the local update thread to record the time to compute the local update, the number of triangle pairs searched during the local update, and the time for the global search to compute the LMDs. The following figures show these results for a variety of model-model interactions. In all these examples, the top graph represents the local update time, the middle graph the number of triangle-pairs searched during the local update, and the bottom graph the time for the global LMD computation to update. The local update and searched triangles graphs do not cover the full extent of the global search graph since the data was stored in a circular buffer and the fast updates of the first two graphs filled the available space.

A. Gear-Crank

In this example, we moved a gear with 6,300 triangles around a crankshaft part with 45,000 triangles, exploring the concave regions (Figure 8). Even though the models are high resolution, typically there were only a few LMDs to track, and the local update was able to maintain a high update rate.

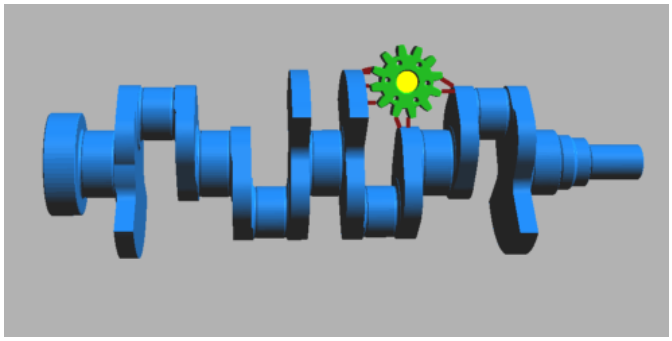

Fig. 8. The gear model is able to explore regions of the crankshaft model.

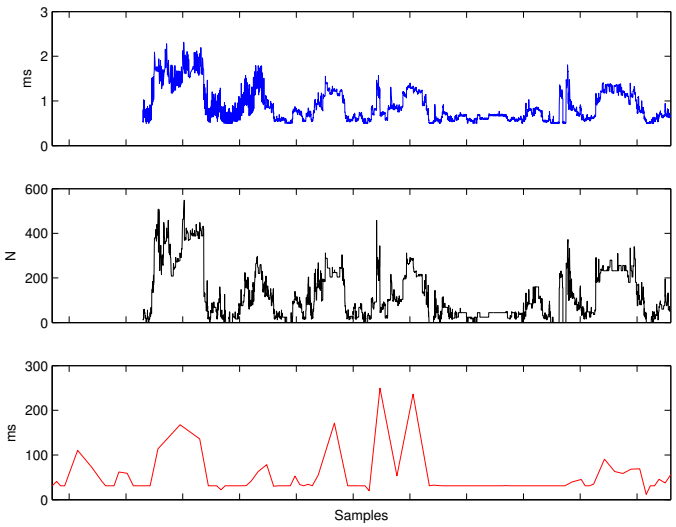

Fig. 9. Update time for the force computation loop (top), number of LMDs (middle), and global LMD update (bottom).

The top graph of Figure 9 shows the time the local update took to update the LMDs and compute forces. The local update was able to maintain near kilohertz rates even during complex interactions. The middle graph counts the number of triangle pairs searched during each local update. The local update time correlates well with the number of triangles searched. The bottom graph shows the computation time for the global search to find the LMDs. Without the local update, haptic interaction would have been highly unstable and slow.

B. Horse-Bunny

Non-mechanical models, such as the horse and bunny (Figure 10), provide additional challenges to our haptic rendering system, as the finely detailed surfaces can produce nearly redundant local minima. In this example, we are still able to update the LMDs and forces at around 1 kilohertz (Figure 11).

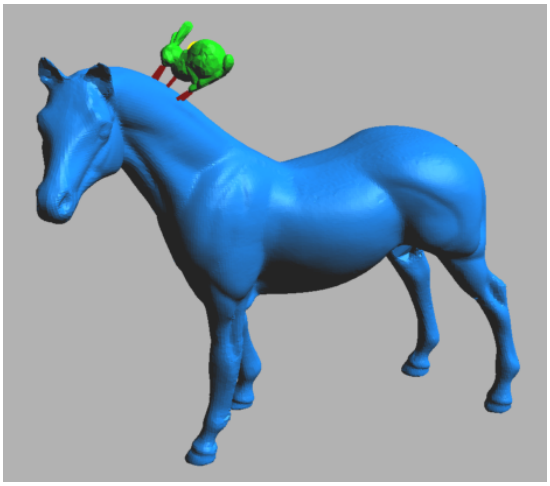

Fig. 10. More detailed models, such as the bunny and horse models, still provide haptic rate performance.

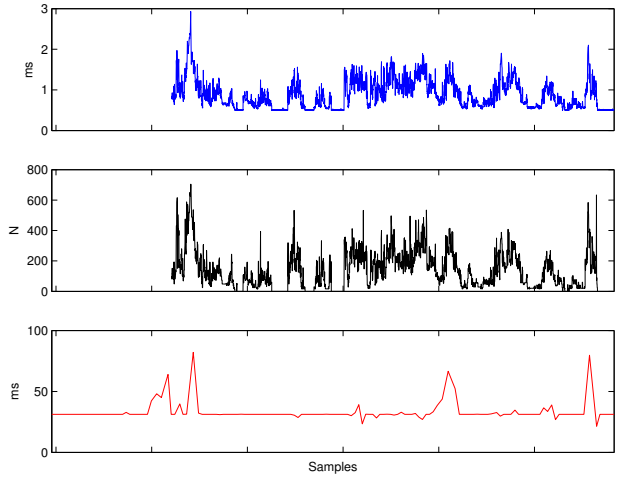

Fig. 11. Update time for the force computation loop (top), number of LMDs (middle), and global LMD update (bottom) during bunny-horse haptic rendering.

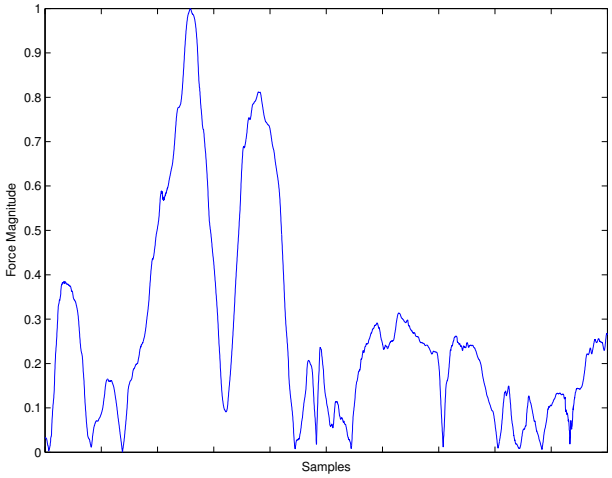

Fig. 12. This graph shows the magnitude of the translation forces for the bunny-horse interaction. While the overall magnitude varies widely, the changes are relatively smooth, indicating stable haptic response.

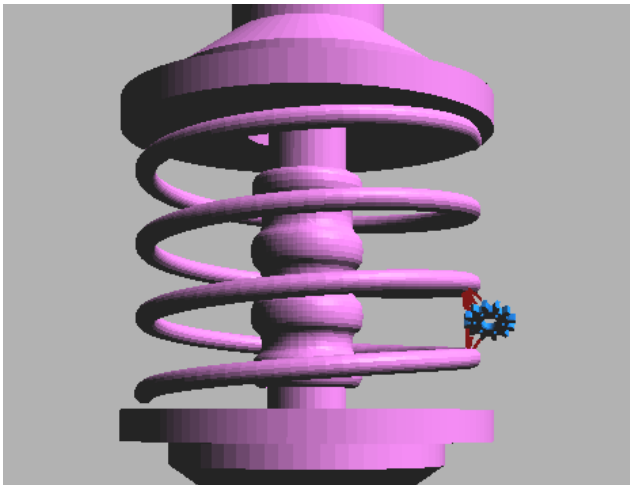

Fig. 13. Forces push the moving model toward a safe path.

The forces of interaction feel smooth. Figure 12 shows the magnitude of the translational forces during haptic interaction between the horse model and the bunny. The large-scale bumps are from moving the bunny model around the horse and bumping against it. Smoother responses are possible when continuously pressing the two models together. However, the lack of high-frequency fluctuations shows good haptic rendering stability.

VIII. An Accessibility Application

Since we compute LMDs while the moving model is still some distance from the environment models, haptic forces are used to guide the moving model away from collision with the environment. The onset distance for forces is adjustable, so the user can decide how much clearance between models is desired during testing. In general, the LMDs tend to approximate the local distance field between the models, and the forces tend to push the moving model toward the medial axis between the models (Figure 13). Since the medial axis is the surface of maximum clearance between models, these forces tend to guide the moving model toward the safest path.

A. Collision-Free Path

While the test object is being moved by the haptic interface, its position and orientation are stored in a buffer. This buffer allows the motion of the test object to be played back for review, analysis, or further modification. If the moving model is forced to penetrate an environment model by the user, the simulation is

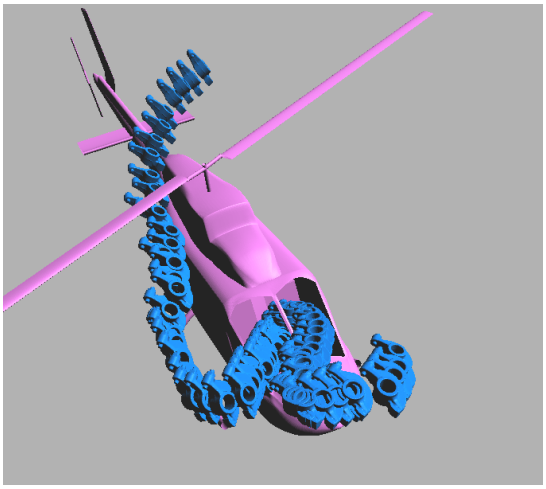

Fig. 14. A user is able to find a path through the cockpit using haptic feedback.

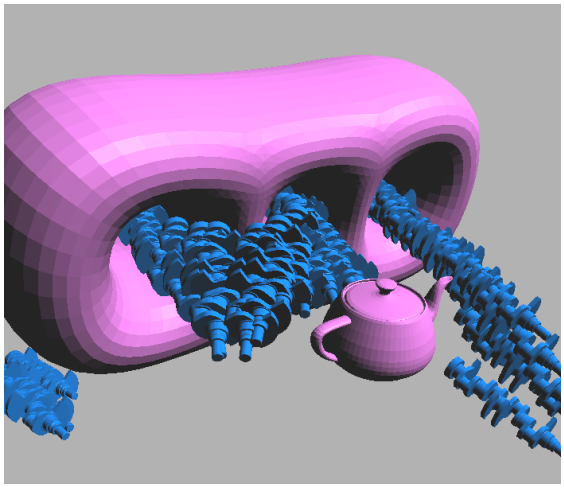

Fig. 15. In this example, the desired path was originally visually occluded, so haptic response provided key cues for moving past the teapot.

no longer valid. A collision state is detected and the simulation is rolled back, using the stored positions and orientations in the buffer, until the model state is valid. The simulation can then resume, and the user can try new approaches for finding a collision-free path. This means that the path stored by our accessibility application is always valid, and if the moving model can reach its goal, the problem has been solved.

B. Detecting Collisions

Collisions are detected when the smallest LMD falls below an adjustable parameter. This parameter can represent error in the fit of the polygonal model to an original CAD model, or it can represent a desired minimum clearance between models. Detecting collisions in this fashion, instead of with actual model intersection, provides more control over simulation accuracy.

C. Path Visualization

Since we store model positions and orientations during the simulation, a sampling of the path of the model can be visualized, as in Figure 14. Drawing a copy of the moving model at each sampled location (or some subset), allows the user to check the validity of the collision-free path, and to examine any unusual maneuvering needed to safely guide the model. One drawback is that if many positions are visualized simultaneously, the frame rate of the display can slow.

D. Examples

Figures 14 and 15 show two challenging examples of trying to find a collision-free path. In Figure 14, a mechanical model with 40,000 triangles is maneuvered through the open cockpit of a 113,000 triangle helicopter model. In Figure 15, the user had to manipulate the crankshaft model through the holes and past a visually occluding teapot model using the haptic interface. The rotated path visualization view shows the forces guided the moving model away from contact.

IX. Penetration Estimation Using SNCH Search

The minimum translational distance (MTD) is the smallest movement needed to separate two convex models. This measure has been used for 6-DOF haptic rendering in [18]. Decompositions of general models into convex surface patches, and subsequent clustering of MTDs for force computation is developed in [19] and [20]. An alternative approach is to find the extremal distance [24], which has been applied to haptic rendering of spline models in [5].

We adapt the SNCH search to extremal distance and use the resulting polygonal model-model local extrema as a penetration measure (Figure 16). While the pruning test for local minima compared the solution line cone and node normal cones looking for normals that pointed in toward each other, for extremal distance we just need the normal cones to point away from each other. This is accomplished by switching the π and 0 in Eq. 5.

However, the leaf test is not as simple to adapt, since the first step of that test is to find the closest pair of points on the leaf triangles. An extremal distance test would require that we instead find all overlaps of normal ranges and then determine the geometry that fulfills the collinearity requirement.

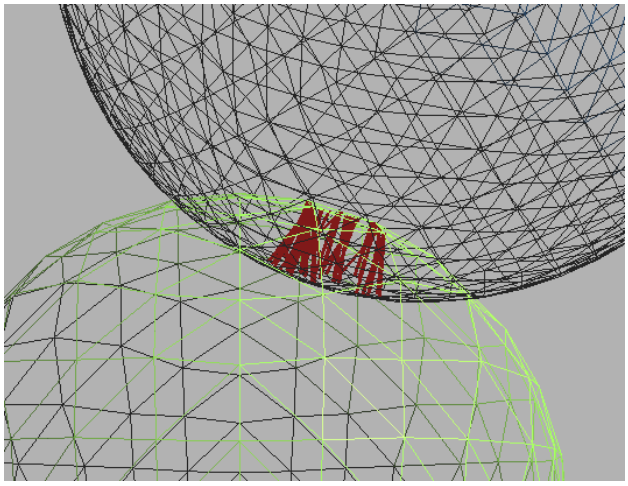

Fig. 16. The penetration depth is approximated by sets of surviving active pairs.

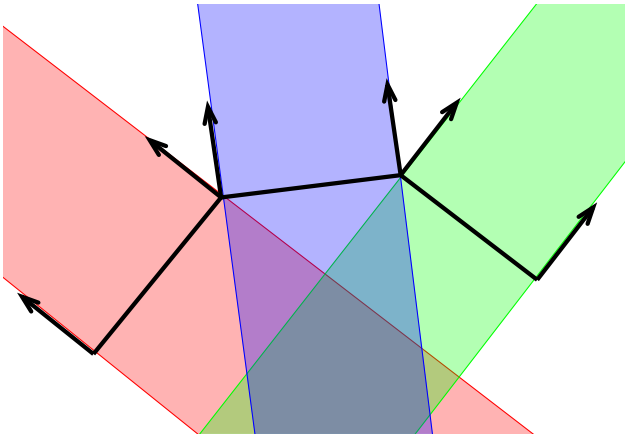

Fig. 17. The normals associated with the faces and edges of a model overlap on the interior of a convex object. These overlaps create multiple solutions for extremal distance on triangulated models.

Furthermore, the generic case of two convex portions of a model overlap produces many local extrema for polygonal models. Since a polygonal model has multiple overlapping normal ranges on the concave side of a model (Figure 17), the interplay of two concave sides produces numerous valid local extrema solutions.

We use the approach of accepting leafs based just on the normal cone tests, rather than doing extra computation that may not yield much additional pruning. Points on each triangle are still needed to find an actual solution line. We use the closest pairs of points on the triangles to form the solution line even in the extremal case.

The main reasons using the closest points are

- For high-resolution models, the penetration depth is larger than the triangle size, so shrinking the solution line by some percentage of the triangle size does not affect the overall penetration length substantially.
- During tangential model contact, using the closest points keeps the solution line zero length, while other heuristics such as connecting triangle centers would introduce spurious torques.

A. *Adaptive Cutoff Distance*

We introduce adaptive cutoff distances to increase the computational efficiency of the penetration depth search. In prior sections, the cutoff distance was fixed based on where the onset of forces was desired for the LMD scenarios or based on the estimated maximum penetration depth in the penetration case. However, given a set of penetration depths at one instance, the maximum penetration depth cannot grow by more than the relative movement of the interpenetrating models. The relative movement is bounded by the incremental translation of the moving model plus the translation of a corner of an oriented bounding box surrounding the model after going through the incremental rotation for that time step.

Therefore, at each time step, the maximum penetration depth from the last time step is added to the bounded relative movement of the models, and this value is used as the distance cutoff for the new time step. This approach has two advantages. It provides a faster response after initial contact, when the penetration depth is small, which improves the quality of the haptic interaction. It also allows the penetration depth to be unbounded, whereas for a fixed cutoff distance, forces will disappear if the models penetrate too far.

B. *Results and Examples*

The penetration depth approach cannot handle as high-resolution models as the LMD approach because it lacks the local search updates and because more active pairs are typically retained than for minimum distance (Figure 18). Figure 19 shows that the penetration depth approach typically runs at a few hundred Hertz. In practice, it provides stable and solid-feeling haptic rendering of the model-model interaction.

The adaptive cutoff distance technique improves the search time for penetration depth estimation. In Figure 20, the search time is recorded as two sphere models were slowly pushed together and then pulled apart, corresponding to a rise in computation time and then a fall. Note that at initial contact, the computation times indicate kilohertz rate feedback. Without the adaptive cutoff distance, the time would have stayed nearer the worst-case time for all penetration depths.

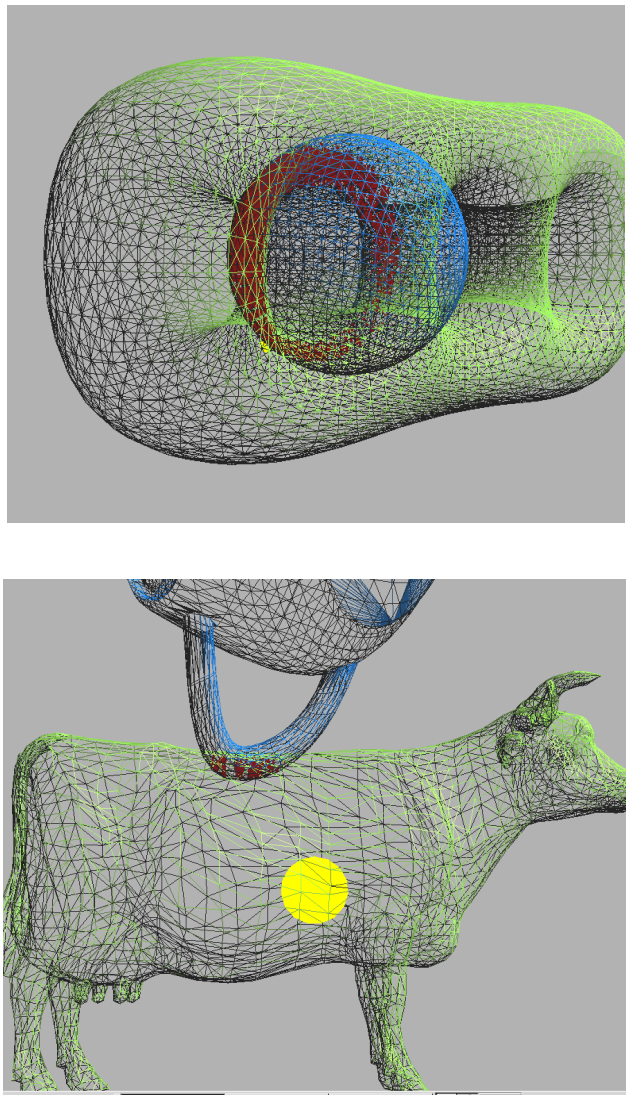

Fig. 18. The sphere is able to explore a hole in the model with penetration forces generated all along the boundary. In the lower figure, the handle of the teapot is kept from penetrating far by the haptic interface. In both cases, more active pairs are used for force computation than typically are used in minimum distance scenarios.

X. Conclusion

We demonstrate an algorithm for six DOF haptic rendering of arbitrary polygonal models. The underlying distance search computes local minimum distances between models and derives repulsive forces and torques to maintain collision-free status. This technique is appropriate as a foundation for an accessibility application for complex models. In addition, a variation in the local minimum search estimates the extremal distance between models, and these extrema are used to compute forces for interpenetrating models. The combined approaches provide powerful tools for adding haptic cues to complex virtual environments.

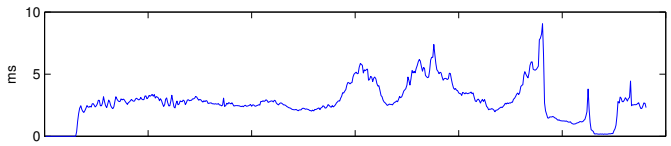

Fig. 19. The penetration depth search for the sphere/3-hole-torus example runs at a few hundred Hertz. The sphere has 8192 triangles and the torus has 11,776 triangles. The spikes in the time are during sphere-hole contact, where many more active pairs are retained.

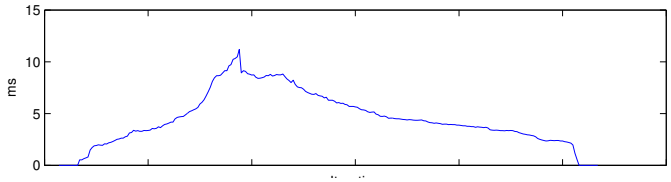

Fig. 20. Timings were collected as two sphere models were pushed together and then pulled apart. The adaptive distance cutoff produces higher computation rates when the penetration depth is small.

XI. Acknowledgments

The authors would like to acknowledge support in part from the following grants: NSF IIS-0428856, NSF CDA-96-23614, and ARO DAAD 19-01-1-0013. Also, we would like to thank the Gamma group at UNC for making their code available.

References

[1] S. Quinlan, "Efficient distance computation between non-convex objects," in *IEEE International Conference on Robotics and Automation (ICRA)*, 1994, pp. 3324–3329.

[2] M. Ponamgi, D. Manocha, and M. C. Lin, "Incremental algorithms for collision detection between solid models," in *ACM/SIGGRAPH Symposium on Solid Modeling*, 1995, pp. 293–304.

[3] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha, "Fast distance queries with rectangular swept sphere volumes," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 4, April 2000, pp. 24–48.

[4] S. A. Ehmann and M. C. Lin, "Accurate and fast proximity queries between polyhedra using convex surface decomposition," in *Eurographics (EG) 2001 Proceedings*, A. Chalmers and T.-M. Rhyne, Eds. Blackwell Publishing, 2001, vol. 20(3), pp. 500–510.

[5] D. Nelson, D. Johnson, and E. Cohen, "Haptic rendering of surface-to-surface sculpted model interaction," in *Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems (HAPTICS 1999)*, 1999.

[6] V. Patoglu and B. Gillespie, "Extremal distance maintenance for parametric curves and surfaces," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[7] D. E. Johnson and E. Cohen, "A framework for efficient minimum distance computations," in *IEEE International Conference on Robotics and Automation*, May 1998, pp. 3678–3684, leuven, Belgium.

[8] D. Johnson and E. Cohen, "Spatialized normal cone hierarchies," in *ACM SIGGRAPH Symposium on Interactive 3D Graphics (I3D)*, March 2001, pp. 129–134.

[9] Y. Adachi, T. Kumano, and K. Ogino, "Intermediate representation for stiff virtual objects," in *Virtual Reality Annual International Symposium (VRAIS'95)*, 1995, pp. 203–210, research Triangle Park, N.C.

[10] A. Gregory, M. C. Lin, S. Gottschalk, and R. Taylor, "A framework for fast and accurate collision detection for haptic interaction," in *IEEE Virtual Reality '99*, 1999, pp. 38–45.

[11] D. Ruspini, K. Kolarov, and O. Khatib, "The haptic display of complex graphical environments," in *Computer Graphics and Interactive Techniques (SIGGRAPH 1997)*, August 1997, pp. 345–352.

[12] K. Salisbury, D. Brock, T. Massie, N. Swarup, and C. Zilles, "Haptic rendering: Programming touch interaction with virtual objects," in *ACM SIGGRAPH Symposium on Interactive 3D Graphics (I3D)*, April 1995, pp. 123–130.

[13] C. Ho, C. Basdogan, and M. Srinivasan, "An efficient haptic rendering technique for displaying 3d polyhedral objects and their surface details in virtual environments," *PRESENCE: Teleoperators and Virtual Environments*, vol. 8, no. 5, pp. 477–491, 1999, mIT Press.

[14] K. Salisbury and C. Tarr, "Haptic rendering of surfaces defined by implicit functions," in *ASME Dynamic Systems and Control Division*, vol. 61, 1997, pp. 61–67.

[15] P. Stewart, "Cad data representations for haptic virtual prototyping," in *ASME Design Engineering Technical Conferences (DETC'97)*, 1997.

[16] T. V. Thompson II and E. Cohen, "Direct haptic rendering of complex trimmed nurbs models," 1999.

[17] D. Baraff, "Fast contact force computation for nonpenetrating rigid bodies," *Computer Graphics*, vol. 28, no. Annual Conference Series, pp. 23–34, 1994.

[18] A. D. Gregory, A. Mascarenhas, S. Ehmann, M. C. Lin, and D. Manocha, "Six degree-of-freedom haptic display of polygonal models," in *IEEE Visualization 2000*, T. Ertl, B. Hamann, and A. Varshney, Eds., 2000, pp. 139–146.

[19] Y. Kim, M. Otaduy, M. C. Lin, and D. Manocha, "Six degree-of-freedom haptic display using localized contact computations," in *Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems (HAPTICS 2002)*, March 2002, pp. 209–216.

[20] M. A. Otaduy and M. C. Lin, "Sensation preserving simplification for haptic rendering," *ACM Transactions on Graphics (TOG): SIGGRAPH 2003*, vol. 22, no. 3, pp. 543–553, 2003.

[21] W. McNeely, K. Puterbaugh, and J. Troy, "Six degree-of-freedom haptic rendering using voxel sampling," in *Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1999)*, 1999, pp. 401–408.

[22] D. E. Johnson and P. Willemsen, "Six degree-of-freedom haptic rendering of complex polygonal models," in *Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems (HAPTICS 2003)*, March 2003, pp. 229–235.

[23] D. Johnson and P. Willemsen, "Accelerated haptic rendering of polygonal models through local descent," in *Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems (HAPTICS 2004)*, March 2004, pp. 18–23.

[24] D. Baraff, "Curved surfaces and coherence for non-penetrating rigid body simulation," *Computer Graphics*, vol. 24, no. 4, pp. 19–28, 1990.