Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2007)
D. Metaxas and J. Popovic (Editors)

Time-critical distributed contact for 6-DoF haptic rendering of adaptively sampled reduced deformable models

Jernej Barbič[1] and Doug L. James[2]

[1]Carnegie Mellon University, Pittsburgh, PA, USA, [2]Cornell University, Ithaca, NY, USA

Abstract

Real-time evaluation of distributed contact forces for rigid or deformable 3D objects is important for providing multi-sensory feedback in emerging real-time applications, such as 6-DoF haptic force-feedback rendering. Unfortunately, at very high temporal rates (1 kHz for haptics), there is often insufficient time to resolve distributed contact between geometrically complex objects.
In this paper, we present a spatially and temporally adaptive sample-based approach to approximate contact forces under hard real-time constraints. The approach is CPU based, and supports contact between a rigid and a reduced deformable model with complex geometry. Penalty-based contact forces are efficiently resolved using a multi-resolution point-based representation for one object, and a signed-distance field for the other. Hard real-time approximation of distributed contact forces uses multi-level progressive point-contact sampling, and exploits temporal coherence, graceful degradation and other optimizations. We present several examples of 6-DoF haptic rendering of geometrically complex rigid and deformable objects in distributed contact at real-time kilohertz rates.

Categories and Subject Descriptors (according to ACM CCS): I.6.8 [Simulation and Modeling]: Types of Simulation: Animation, I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling: Physically based modeling, I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism: Virtual Reality

1. Introduction

In this paper, we address *distributed* contact between geometrically complex 3D objects, i.e., contact with potentially several simultaneous contact sites each distributed over a non-zero surface area (also called multi-point contact; see Figure 1). Distributed contact represents an important component of interactive physically based virtual environments. It is challenging to simulate due to competing demands of having interesting geometrically complex rigid and (more notoriously) deformable objects, as well as high real-time contact force calculations rates, e.g., 1 kHz to support haptic force-feedback rendering. We propose an efficient CPU-based algorithm suitable for hard real-time evaluation of distributed contact forces between geometrically complex models. These real-time forces can be used in a variety of applications, however, we focus on the challenging application of 6-DoF haptic rendering of rigid and deformable models.

Our approach is most closely related to the Voxmap-PointShell method (VPS) [MPT99], a 6-DoF haptic rendering algorithm capable of simulating distributed geometrically detailed contact between a *rigid* environment and a *rigid* point-sampled object (*pointshell*). The pointshell is a collection of points (with normals) positioned near-regularly at the surface of the object. The VPS environment is modeled as a *voxmap* datastructure: a voxelization of the polygonal environment geometry where each voxel contains a fixed-width (e.g., 2-bit) value (e.g., interior, surface, next to surface, free space). In every VPS haptic cycle, pointshell points are queried against the voxmap to determine contact.

Unfortunately, objects modeled by VPS and most notable recent works on 6-DoF haptic rendering are entirely rigid, and for good reason: nonlinear deformable objects are computationally expensive since (a) detailed finite element simulations can not easily achieve haptic rates (1000 Hz) for geometrically interesting models, and (b) deformable collision detection is expensive especially for close-proximity scenarios typical of contact. These fundamental difficulties have seriously hampered the progress of a variety of emerg-

SCA 2007, San Diego, California, August 04 - 05, 2007
© 2007 ACM 978-1-59593-624-4/07/0008 $ 5.00

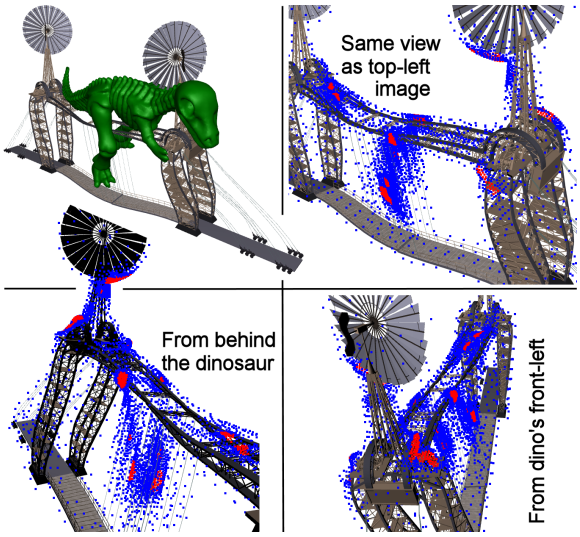

Figure 1: ***Distributed deformable contact:*** *(**Top Left**) contact configuration between a nonlinear reduced deformable bridge and a rigid dinosaur, with 12 separate contact sites; (**Other images**) contact sites viewed from three different camera angles. A total of 5200 points (in blue) were traversed, resulting in 426 points (in red) in contact. Note: temporal coherence was disabled for this figure.*

ing haptic rendering applications, such as haptic gaming. One other application of haptic rendering is virtual prototyping: haptics can allow airplane designers to check if a certain mechanical part can be positioned into (or removed from) designated locations, thereby avoiding critical design flaws before manufacturing the structure, e.g., fire detector removal from an aircraft engine assembly during routine servicing [WM03]. Supporting deformable geometry is important since airplanes include numerous deformable parts such as hoses, wires, plastic parts, sheet metal, etc. Part-removal analysis using only rigid simulations can lead to false reports that a part cannot be (dis)assembled when in reality it can. For example, wires and hoses may be deformed to let an object through a narrow passage (see Figure 2).

1.1. Overview

Our approach is designed to work with a variety of reduced deformable models that support a basic two-step simulation process: (1) a fast timestep of reduced deformable dynamics; (2) fast evaluation of individual deformed surface point positions and normals to adaptively resolve contact. Suitable reduced models include classic linear modal vibration models, reduced nonlinear models, simple low-resolution deformable models with embedded geometry, adaptive multiresolution models, articulated flexible multibodies, etc. The only requirement is that the model's reduced state can be time-stepped at high rates, and its deformed geometry sampled adaptively for time-critical contact force estimation. In our implementation we employ prior dimensional model reduction and Newmark subspace integration techniques for reduced geometrically nonlinear models [BJ05], which are suitable for large deformations with large rotations but small local strain. Such reduced models are based on formal reductions of large-deformation FEM models. They can simulate reduced dynamics of large deformations of geometrically detailed objects at haptic rates, e.g., much less than one hundred microseconds per timestep in our examples. We make extensive use of the low-dimensional shape parameterization for time-critical collision and force processing. Regardless of the particular reduced deformation model used, our approach addresses stable haptic display of distributed contact between two objects *each* with non-trivial geometry.

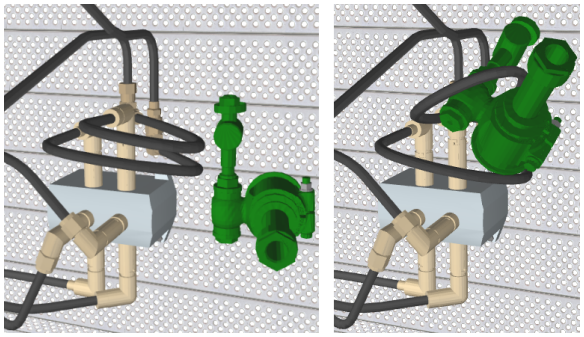

Figure 2: ***Real-time 6-DoF haptic rendering*** *of rigid and deformable objects in contact. The operator is navigating the green rigid part in between two deformable hoses.*

Instead of a VPS voxmap, we use a floating-point signed-distance field to (a) make the contact force continuous (improving stability), (b) support time-critical contact estimation, and (c) exploit temporal coherence. We support contact between a rigid and a reduced deformable model using a rigid distance field and a deformable pointshell (see Figure 3). Output-sensitive signed-distance field oracles for reduced deformable models can be used to support defo-defo contact [Bar07], but are beyond the scope of this paper. Our approach scales with increasing geometric detail, and is (to the best of our knowledge) the first approach that can support haptic rendering of deformable contact between such geometrically complex models.

To support haptic display of geometrically detailed models, e.g., involving a million pointshell points, we organize the pointshell into a *nested* multi-resolution hierarchy, by sampling points (and normals) from an appropriately defined *smooth manifold surface*. Our sampling is near-uniform which provides economic pointshells and improves haptic stability. We use a precomputed sphere-tree hierarchy to bound the pointshell points, *with sphere centers located at (non-leaf) pointshell points*. For deformable models, this gives a point-based modification of the Bounded Deformation Tree (BD-Tree) [JP04]. Such a hierarchy permits large pointshells, bigger than what could be examined point by point during one haptic cycle: if a point on some coarse

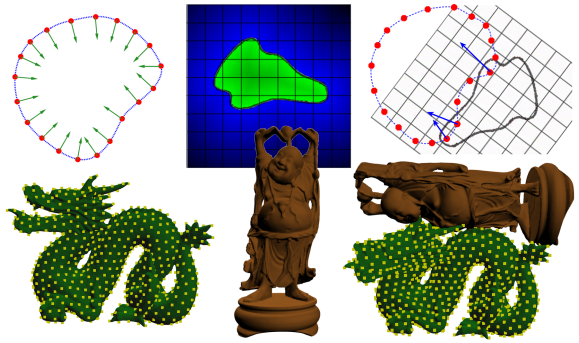

Figure 3: ***Point-contact model:*** *(**Left**) Reduced-deformable pointshell with inward normals; (**Middle**) the distance field; (**Right**) deformable contact and contact forces.*

hierarchical level is sufficiently far from contact, none of the points in the subtree can be in contact, and then those points need not be checked. In contrast, if a large part of the pointshell is in contact or close to contact, few subtrees will be culled and a large part of the pointshell will need to be examined point by point. We address this problem by providing *graceful degradation of contact*: if there is not enough computation time to fully complete the tree traversal, the algorithm still returns a reasonable answer, with accuracy dependent on contact-configuration difficulty and available processing power. Graceful degradation is achieved by traversing the nested hierarchy in a breadth-first order, rendering deeper and deeper tree levels until out of computation time. We use two separate activation thresholds to avoid abrupt changes in the rendered depth during consecutive haptic cycles. The simulation also uses temporal coherence to time-sample the individual points at rates depending on distance to contact (temporal adaptivity).

We haptically display distributed contact using a novel first-order quasi-dynamic virtual coupling method. In haptics, higher update rates generally enable stable rendering of stiffer forces and torques; but reaching such high rates is challenging for models with detailed geometry. In our system, the deformations, collision detection and contact force computations run together in one loop at haptic rates (1000 Hz). We do not need to extrapolate from lower-rate signals. A multi-rate simulation using the contact gradients of Section 6 could exploit user's limited sensing bandwidth to further increase example complexity [OL06, OG07].

1.2. Other Related Work

Haptic rendering has received increasing attention over the last decade [LD07], and significant challenges remain especially for rendering deformable systems. Early methods were developed for tissue simulations during virtual surgery [KcM00], and mostly used mass-spring deformable objects. Space-time adaptive and multi-rate methods have been successful for real-time deformable object simulation [AH98, DDCB01], but time-critical distributed contact algorithms have received less attention (c.f. [MO06]). A variety of preprocessing techniques, such as precomputed Green's functions, exist for accelerating linear elastostatic models to support haptic force-feedback rendering of point-like contact, or modest multi-point contacts via low-rank updates [BNC96, CDA99, JP01, JP03, JCC06]. Unfortunately, most such haptic rendering papers assume interactions based on pulling on (or constraining) certain vertices, not performing real-time deformable-rigid object collision detection and distributed contact force and torque computations with complex geometry.

As mentioned, the Voxmap-PointShell (VPS) method was introduced by [MPT99] to support rigid-rigid contact rendering. Pointshell accuracy and virtual coupling stability were improved in [RPP*01]. In [WM03], they replaced the original VPS mass-spring-damper with a quasi-static spring. The recent paper of [MPT06] introduces wider voxmaps (4-bit), which now have more room to store distance-to-nearest-contact information for free-space voxels. Since the haptic device is user controlled, there exists an upper bound on the maximum possible velocity of a point shell point, and therefore points located in a voxel sufficiently far from contact need not be checked for several haptic cycles (temporal coherence). Also, the pointshell is traversed on two levels: a coarse level shell is tested against a coarse level voxmap, followed by finer tests if necessary. The hierarchy only consists of two levels, the simulation is rigid, and graceful degradation of contact was not addressed.

Our progressive interruptible point-based contact-sampling approach is analogous to the QSplat point-based rendering algorithm [RL00] which represented the point cloud hierarchically to enable level-of-detail control for *graphical* rendering. Our oriented pointshell can be seen as a surfel point cloud. A single-resolution contact algorithm for two mesh-free deformable point-sampled objects has been presented in [KMH*04]. The method computes the contact surface using the point-based surface projection operator, reaching interactive (but not haptic) rates: timesteps of 130-200 milliseconds were reported for two models of about 10,000 surfels each. Hierarchical point-contact handling was explored in [PPG04], where they demonstrate near-interactive (a few frames per second without precomputation) contact of point-based quasi-rigid (quasi-static and linear) deformable objects. In this paper, we present a hierarchical contact rendering method designed for high-rate (haptic rendering) applications. We demonstrate how to maintain upper bounds on the amount of computation per haptic cycle, and how to minimize contact force artifacts when run-time constraints force a transition to a different contact resolution level.

Bounding volume hierarchies can be used for *interruptible* collision detection for rigid [Hub95] and embedded deformable models [MO06] by traversing the hierarchy in

a breadth-first manner. Time-critical contact can be approximated by applying contact forces between contacting (sphere) primitives at the deepest level traversed in a particular frame. Unfortunately, such approaches can have trouble resolving close-proximity and conforming contact configurations in high-rate scenarios (as in haptic rendering): finer bounding volumes more closely approximate the object, and as such the effective contact surface (and therefore forces) depend on the level of detail rendered at any instant. In our work, the points at different resolution levels progressively sample the same well-defined contact surface, thereby making contact at different levels of detail more consistent.

Six-DoF haptic rendering of complex *rigid* objects can be achieved using a number of techniques, in addition to VPS, such as multi-resolution collision detection on convex decompositions, followed by a clustering of contacts [KOLM03]. Multi-resolution contact haptics has been demonstrated for polygonal geometry by appropriately simplifying the objects at different hierarchical levels, to preserve the user haptic *sensation* [OL03, OL06]. The authors suggest that for contact sites of large area, resolving the contact at a coarser resolution will not result in significant sensation degradation; this has motivated our graceful degradation algorithm. These approaches, however, do not directly apply to point-sampled objects or deformable objects.

Linear complementary problem (LCP) solvers can be used to solve the Signorini contact problem between a rigid object and a linear deformable object [DDKA06]. LCP solvers are computationally demanding which limits the approach to models of modest geometric complexity. A deformable pointshell haptic rendering example has been demonstrated in [BJ05]; however, one of the two contacting objects (a small rigid ball) had trivial geometry. In this paper, *both* contacting objects have non-trivial geometry, resulting in a much more difficult collision detection problem; also the contact model of analytical gradients (as used with the ball) gives discontinuities if naively extended to non-trivial distance field geometry.

2. Penalty-based Point-contact Model

The point-contact model employed for rigid-rigid, or rigid-defo (or even defo-defo) contact are conceptually identical: the contact penalty forces are determined by querying the points of the pointshell object against the signed distance field of the other object (see Figure 3). The user interactively manipulates one of the two objects (the *haptic object*); in our implementation, the distance-field object is manipulated.

Both pointshell and the distance field are computed in a pre-processing stage. For completeness, our particular distance field preprocess is described in Appendix A. During every haptic cycle, pointshell points are queried against the signed-distance field, as illustrated in Figure 3. By convention, negative distance field values denote the interior of the distance field object, and therefore points in contact. Penalty forces are computed for points in contact. We choose the direction of the force to equal the inward normal of the pointshell point in contact (and as such the force acts to resolve contact; see also Section 4). The magnitude is determined from the amount of penetration: $F = -k_C \cdot d \cdot N$, where k_C is the contact penalty force stiffness, $d < 0$ is the signed distance field value, and N is the point's inward normal in the world coordinate system. The point world-coordinate position and normal computation are discussed in later sections. Note that the total force and torque can be computed easily using this model, and the trilinearly interpolated distance field.

We note that this contact model is not standard, but it has useful approximation properties for haptics. It avoids using the distance field gradients which are discontinuous across the object's interior medial axes. In fact, we first tried using distance field gradients to determine the direction of the penalty force, but the resulting force direction discontinuities gave poor haptic signals. In our contact model, the force on a point is a continuous function of objects' position and deformation. Only the normal of one of the two contacting objects (that of the pointshell object) determines force direction, and this is both an approximation used for speed, and robustness; also note that for smooth surfaces in non-penetrating contact, the normals of the two opposing surfaces at the contact point are colinear.

Comparison: We also implemented the standard VPS voxmap [MPT99] and found distance fields to provide more stable haptic feedback, especially at higher stiffness settings. We attribute this difference to discontinuities in voxmap contact force when crossing voxel boundaries. To reduce distance-field memory, a hybrid data structure would be possible, using a wide-bit voxmap in free space, and a distance field accessed via a hash table in the shallow contact force layer surrounding the object.

3. Sampling reduced deformable models

Without loss of generality (among reduced models) we assume the reduced deformation kinematics $u(t) = U \cdot q(t)$, where $u \in \mathbb{R}^{3n}$ is the displacement vector (e.g., for a FEM simulation mesh with n vertices), $U \in \mathbb{R}^{3n \times r}$ is a certain time-independent low-rank deformation basis matrix, and $q = q(t) \in \mathbb{R}^r$ is the vector of reduced coordinates. The key algorithmic idea is that once the reduced state, q and $\dot{q}$, has been updated by the reduced integrator, co-state quantities such as individual deformable point positions and normals can be sampled adaptively as needed by the tree traversal.

Point samples: Given the current reduced coordinates q of any haptic cycle, any of the deforming pointshell's S points can be reconstructed in $3r$-flops by summing up the r per-

vertex displacement modes:

$$p_s = P_s + U_s\, q, \quad s = 1 \ldots S, \tag{1}$$

where P_s is the undeformed pointshell position, and $U_s \in \mathbb{R}^{3\times r}$ is the precomputed modal displacement matrix (interpolated at P_s from U using the FEM shape functions). If the point is in contact ($d(p_s) < 0$), the normal will be computed, and the penalty contact force evaluated; this force $f_s \in \mathbb{R}^3$ is then applied as an external reduced force $U_s^T f_s \in \mathbb{R}^r$ to the reduced deformable object integrator to excite deformations. This integration timestep is performed at the end of the haptic cycle, producing q and $\dot{q}$ for the next haptic cycle.

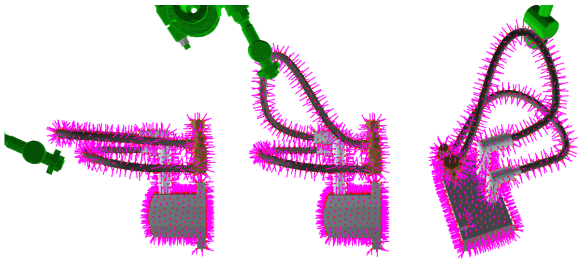

Figure 4: ***Our dynamic normals:*** *(**Left**) undeformed; (**Middle, Right**) deformed, viewed from two camera angles. In practice, normals need only be computed for pointshell points in contact. Maximum deviation from the correct surface normal is 5.2 degrees in this example; and less than 3.6 degrees for all but five points.*

Fast normal approximation: Normals must be computed for sparse pointshell points *in contact* for rapid evaluation of penalty forces. Rigid pointshells have static body-frame normals that can be precomputed. Deforming pointshell normals can be approximated in a variety of ways. In our implementation, we exploit the fact that our geometrically nonlinear deformable models involve large rotations but small strain. This enables us to well approximate the deformed normals simply by transforming the undeformed normal by the local deformation gradient, followed by normalization (see Figure 4). The deformation gradient at P_s ($\frac{\partial}{\partial P_s}$ of (1)), interpolated from the FEM model, is a linear function of q,

$$F_s(q) = I + \sum_{j=1..r} F_{s,j}\, q_j, \tag{2}$$

where s is the index of the pointshell point, and $F_{s,j} \in \mathbb{R}^{3,3}$ can be computed from the deformation basis matrix U and the FEM shape functions. This allows us to construct a fast $3r$-flop pointshell normal approximation,

$$n_s = \frac{n'_s}{\|n'_s\|}, \quad n'_s \approx F_s(q)\cdot N_s = N_s + \Delta N_s\, q, \quad s = 1 \ldots S, \tag{3}$$

where N_s is the undeformed normal, and $\Delta N_s \in \mathbb{R}^{3\times r}$ is a precomputed normal-correction matrix (whose j-th column equals $F_{s,j}\cdot N_s$) that accounts for large rotations–unlike the linearized approximation in [JP02]. Finally, at double the cost, true normals could be sampled by instead transforming the tangent space.

4. Nested Pointshell Hierarchy

Only small pointshells ($\sim$4000 points on our hardware) can be traversed linearly (point by point) in a haptic cycle. To some extent, larger pointshells are possible by using temporal coherence and wider voxels [MPT06]. To be able to simulate much larger pointshells (rigid or deformable), we organize the pointshell into a *nested tree hierarchy* (see Figure 5). This (deformable) pointshell hierarchy also serves as a (BD-Tree) bounding sphere hierarchy.

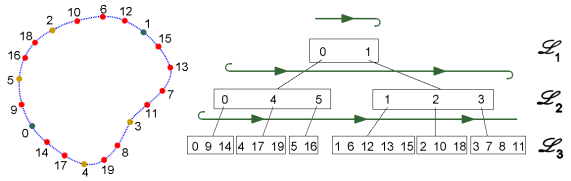

Figure 5: ***Nested point-tree:*** *(**Left**) The pointshell; (**Right**) the hierarchy, the traversal order and tree levels $\mathcal{L}_i$. Particle-repulsion levels are 0-1, 2-5, and 6-19 in this case.*

Uniform pointshell sampling and meaningful pointshell normals are important for providing reliable contact-force estimates and stable haptic rendering. At first, we considered pointshells sampling polygon soup geometry, but stable haptic rendering cannot be guaranteed due to unpredictable normals and the non-closed and non-oriented nature of the surface sampling. Consequently, we designed our pointshells to sample a *closed manifold oriented surface*. When combined with virtual coupling saturation (§ 6), this restricts the distance field object to outside of the pointshell object (or at most a shallow penetration). Another problem happens if points are too far apart, allowing a small feature in the distance field object to slip undetected in between two points; we address this by keeping the pointshell resolution equal or finer than the distance field resolution. In general, we found that pointshells sampling such surfaces tend to be stable; high frequency components, either in point locations or point normals tend to cause instabilities. Sharp corners can still be rendered, albeit not aggressively.

Multi-resolution pointshell via particle repulsion: We generate our pointshells by fitting a set of *particles* (points) onto a smooth manifold surface, using the ideas of particle repulsion [Tur92, WH94]. Similar to [Tur01], we construct a nested point hierarchy by progressively fitting increasing finer samplings of points ($4\times$ branching factor), while freezing coarser level points; we, however, fit points on a level set of a distance field as opposed to a triangle mesh (see Figure 6). Details are given in Appendix B.

Nested pointshell sphere-tree construction: After particle repulsion, each point is positioned, and assigned to one of the disjoint *particle-repulsion* levels. We proceed by organizing the points into a sphere-tree hierarchy. In the deformable case, this results in a point-based Bounded Deformation Tree (BD-Tree). We first generate *tree levels* $\mathcal{L}_i$:

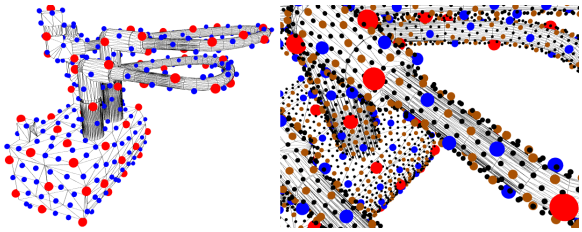

Figure 6: ***Multi-resolution oriented pointshell:*** *(**Left**) The first two levels of a part of the hose scene; (**Right**) detail with four levels shown. Points are fitted on a small-distance offset oriented manifold surface to support "polygon soup" input geometry, and provide oriented surface normals for contact.*

the set $\mathcal{L}_i$ consists of an instance of every point of particle-repulsion levels 1 through i (see Figure 5). Therefore, if a point first appears on particle-repulsion level i, an instance of this point will appear in all tree levels $\mathcal{L}_j$, for $j \geq i$. Instances can share common memory data. The deepest tree level consists of a single instance of all the points in the pointshell. The elements of $\mathcal{L}_i$ are our tree *nodes* on tree level i. This construction was chosen so that all nodes at all levels lie on the actual contact surface $\mathcal{T}$, and that each $\mathcal{L}_i$ samples $\mathcal{T}$ about uniformly (without gaps), with progressively finer resolution with level index i. We establish the tree parent-child relationships by traversing nodes at every level $\mathcal{L}_{i+1}$, and assigning each node to the nearest node from level $\mathcal{L}_i$. A bounding sphere radius is computed for each non-leaf tree node, centered at the location of the node, and covering all nodes in the subtree, in the undeformed configuration. The radius of the bounding sphere gets progressively smaller for instances of the same point at different levels. For deformable pointshells, the sphere radius is updated at runtime using appropriate precomputed BD-Tree datastructures (update cost is r flops per traversed node). The sphere center always coincides with the current node location.

5. Time-critical Progressive Contact Forces

We now describe the core approach for time-critical evaluation of contact forces when one object is represented as a rigid or deformable pointshell, and the other object provides a signed-distance oracle, $d(p)$.

Runtime tree traversal: At every haptic cycle, the algorithm traverses the pointshell hierarchy in breadth-first order. For every tree node, the algorithm looks up the value of the signed distance field at the node's location. Next, to support deformations, the BD-Tree method is used to update the radius of the bounding sphere covering points in the subtree rooted at the node. If the distance field value is greater than the radius, no point in the subtree can collide, and the subtree is not traversed further (see Figure 7 (Middle, e.g., node 2)). Otherwise, all children of the node are added to a list for future traversal (see Figure 7 (Middle, e.g., nodes 0, 3)). If the distance field value is negative (contact), a penalty force is assigned to the node, just like in the single-resolution case. If child copies of the node are visited during later traversal, no additional force is accumulated (but the copy is not skipped from traversal since its proper children may be in contact).

The subtle but algorithmically key point is that our point-based BD-Tree uses nested pointshell points as deformed sphere centers. This seemingly trivial choice enables time-critical testing of the point-based BD-Tree against a distance field: updating a BD-Tree sphere also updates a deformed pointshell point, p, which can in turn be immediately evaluated against the distance field, and if in contact ($d(p) < 0$) will contribute to the progressive accumulation of contact forces in a multi-resolution manner. Therefore, even in the expected case where there is insufficient time to completely test the BD-Tree against the distance field, approximate contact forces can still be accumulated in a time-critical fashion.

Temporal coherence: When a tree node is processed, an estimate of the earliest possible haptic cycle when the bounding ball at this node can enter contact is determined (and cached with the node), based on node's current distance to contact, sphere radius, and maximum point velocity that can occur in the simulation. This can be the immediate next cycle if the node is close to contact (or in contact), or it can be hundreds of cycles for points far from contact. Next time a node is about to be processed, we can safely skip it if the current haptic cycle is younger that what stored in the node. Temporal coherence reduced our simulation times 20%-45%.

Graceful degradation: If the contact scenario is not very involved, much of the tree will be quickly culled and only a small fraction of the pointshell will be traversed. In contrast, if there are many different contact sites, such as when the pointshell object is positioned close to the distance field object, a large fraction of the pointshell will need to be traversed, preventing a complete tree traversal within one haptic cycle. To support graceful degradation in such cases, we traverse the tree progressively level by level (see Figure 7). While traversing the list of nodes that need to be visited on level i (as determined by level $i-1$), we build the traversal list for level $i+1$. At the end of level i, we compare the size of this list to a measure of the remaining computation time for the haptic cycle. Only if enough time remains, we render the next level. The algorithm always either completes a level or does not start it. The resulting contact forces are identical to those obtained under an exhaustive tree-less traversal of points up to a given level. Note that such a scheme is of course approximate: only geometric features resolved by points on the deepest traversed level will be rendered. The benefit of our construction is that rendered surface resolution and contact force accuracy both increase with available computing power.

LOD control: The amount of work required to process a node is predictable and it is not necessary to time the execution explicitly. Instead, we count the total number of tree

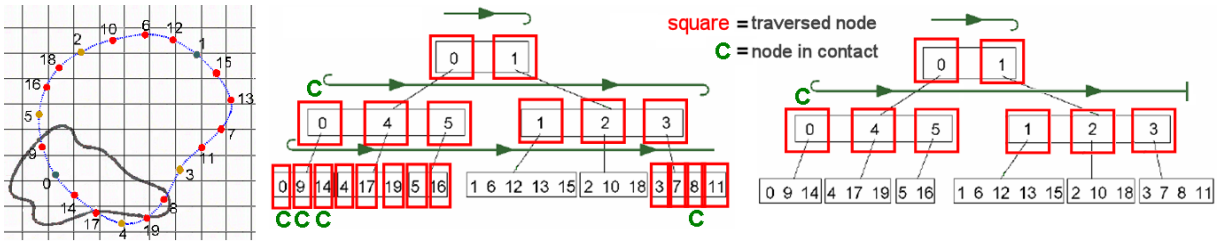

Figure 7: ***Graceful degradation:*** *With involved contact scenarios (**Left**), many tree nodes need to be traversed during a complete hierarchy traversal (**Middle**). Under graceful degradation (**Right**), tree is traversed only up to a shallow depth, as permitted by available CPU power.*

nodes visited, V, during the current haptic cycle. We use two thresholds to determine whether to render the next tree level. All the levels up to and including the deepest level rendered in the previous haptic cycle use the *warm threshold* V_W, whereas all deeper levels use the *cold threshold* $V_C < V_W$. A level is rendered if the total number of tree nodes visited before entering this level is below the level's threshold. Cold threshold is stricter to prevent popping back and forth among two levels during consecutive haptic cycles; we set $V_C = 0.8V_W$. In practice, we have seen LOD changes at most about 2-3$\times$ per second, and did not feel any popping haptic rendering artifacts. LOD changes can be made fewer by reducing V_C. We selected V_W manually by running the simulation and observing computation times per haptic cycle. The selection was not difficult and could be automated, as the cost to process each node is $O(r)$, establishing a high correlation between the number of traversed nodes and the total haptic cycle computation time.

6. Haptic Display of Distributed Contact

Until now we have abstracted 6-DoF haptic rendering as simply a matter of computing contact forces and torques at high rates. However, penalty forces and torques cannot simply be sent to the haptic device: each point in contact adds to the overall stiffness of the system and if enough points are added, maximum renderable stiffness of the haptic device is easily exceeded. Instead, it is customary to separate the *simulation* position of the haptic object from the position imposed by the haptic manipulandum, and connect the two with a generalized spring(*virtual coupling* [CGSS93]). This penalty spring of stiffness k_{VC} tries to align the simulation position and orientation of the haptic object to the manipulandum. We adopt *static* virtual coupling of [WM03] since it needs a relatively small number of parameters (unlike [MPT99]), and does not need device velocity estimates. We extend static coupling by introducing quasi-static damping. We also include large rotation terms in the static equilibrium equations (details in [Bar07]).

In every haptic cycle, we first read the position and orientation of the haptic device manipulandum. We then compute contact penalty forces and torques (by traversing the nested point-tree), virtual coupling forces and torques, *and their gradients* with respect to the *simulation* position of the haptic object (see Figure 3). The net force and torque on the simulation object, under a (small) incremental displacement Δx and (small) incremental rotation $\Delta\omega \in \mathbb{R}^3$, are then

$$F_{\mathrm{net}} = F_{\mathrm{VC}} + F_{\mathrm{C}} + \left(\frac{\partial F_{\mathrm{VC}}}{\partial x} + \frac{\partial F_{\mathrm{C}}}{\partial x}\right)\Delta x + \left(\frac{\partial F_{\mathrm{VC}}}{\partial \omega} + \frac{\partial F_{\mathrm{C}}}{\partial \omega}\right)\Delta\omega \quad (4)$$

$$T_{\mathrm{net}} = T_{\mathrm{VC}} + T_{\mathrm{C}} + \left(\frac{\partial T_{\mathrm{VC}}}{\partial x} + \frac{\partial T_{\mathrm{C}}}{\partial x}\right)\Delta x + \left(\frac{\partial T_{\mathrm{VC}}}{\partial \omega} + \frac{\partial T_{\mathrm{C}}}{\partial \omega}\right)\Delta\omega \quad (5)$$

where $F_{\mathrm{VC}}, T_{\mathrm{VC}}$ denote current virtual coupling force and torque [WM03], and $F_{\mathrm{C}}, T_{\mathrm{C}}$ denote the sum of all contact forces and torques. We displace the virtual object such that the net force and torque on the object vanish under this first-order model, by solving the non-symmetric 6x6 linear system of Equations 4,5 for $(\Delta x, \Delta\omega)$ using LU decomposition (fast, $\sim 5\mu s$ per solve). Finally, we compute the virtual coupling force and torque with respect to the new simulation position, and render them to the user. Virtual coupling gradients are determined through analytical differentiation of the virtual coupling forces and torques. Gradients of contact forces and torques are computed by treating each contact point as undergoing contact with an infinite halfspace (wall), oriented according to the contact normal (details in [Bar07]).

A simple calculation shows that the maximum stiffness felt by the user if m points are in contact is $1/(1/(m \cdot k_{\mathrm{C}}) + (1/k_{\mathrm{VC}}))$; therefore k_{VC} presents an upper bound on the stiffness ever rendered to the haptic device, regardless of m. Since time is sampled discretely, simulation stability is improved if k_{C} is scaled down when there are multiple points in contact; as in VPS, we replace k_{C} with $10k_{\mathrm{C}}/m$ if $m \geq 10$ points are in contact. The displacement-force relationship of virtual coupling is designed to be linear only up to a certain displacement, after which it saturates to some maximum force value (c.f. virtual proxy [ZS95, RKK97]).

Quasi-static damping: The lack of dissipation in the static virtual coupling model can lead to slight instabilities, such as during fast sliding contact. We augmented the static virtual coupling model by introducing a quasi-static damping, or a damped state update: after $\Delta x, \Delta\omega$ are computed, we only change the simulation position and rotation by $(1-\alpha)\Delta x, (1-\alpha)\Delta\omega$, where $\alpha \in [0,1)$ controls the amount of static damping. Such damping exponentially restores the simulation position to the manipulandum position, similar to critical damping in mass-spring-damper dynamic virtual

Contact case; pointshell+distance field	Contact depicted at:	Pointshell		Dist. field	Traversed nodes		Points in	LOD	Timings [μs]				
		levels	points	resolution	with TC	TC off	contact		VC	StVK	TT	total	TL
knight(r) + axe(r)	Results	6	1.02 M	256	2050	3400	820	6	90	0	780	870	80,000
bridge(d) + dinosaur(r)	Figure 1	4	85K	256	1989	5200	426	4	90	45	710	845	12,200
CAD scene(d) + metallic part(r)	Figure 2	4	80K	128	530	1520	50	4	90	45	125	260	6,600
dragon(d) + Buddha(r)	Figure 3	5	256K	256	1750	2900	650	5	90	45	615	750	29,800
dragon(r) + dragon hole(r)	Figure 9	5	256K	256	1701	2642	843	2	90	0	722	812	45,000

Table 1: ***Model statistics*** *for representative contact configurations. Timings generated using an Intel Core 2 Duo 2.66 GHz processor with 2 Gb RAM. All deformable models use $r = 15$ deformation modes for comparative purposes. LEGEND: r=rigid, d=deformable, VC=virtual coupling and manipulandum transformations, StVK=deformable FEM dynamics, TT=tree traversal, TC=temporal coherence, LOD=level of detail rendered (in the particular configuration), TL=time under a tree-less pointshell traversal.*

coupling models [MPT99]. We set $\alpha = 0.5$ in all our simulations; high values of α are avoided since they introduce a surface stickiness effect.

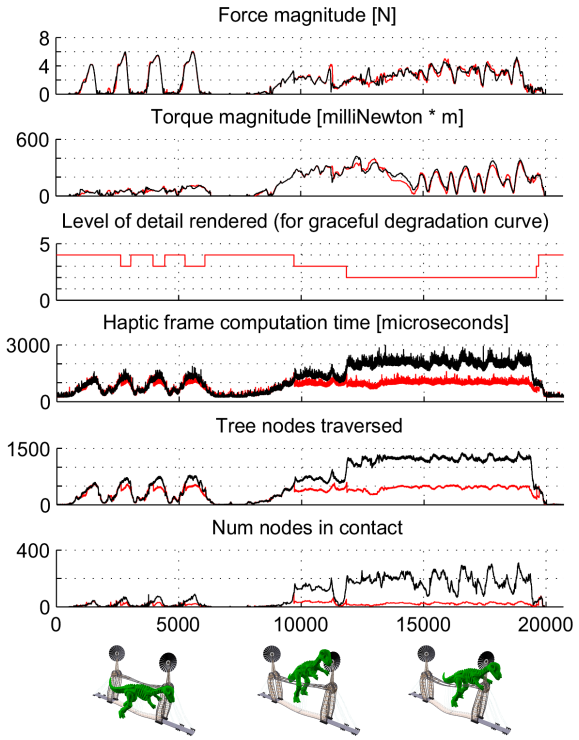

Figure 8: ***Illustration of graceful degradation (GD):*** *Simulation data for a pre-recorded manipulandum trajectory is rendered in two ways: using GD (in red) with warm threshold at 600 nodes; without GD (in black) with all four tree levels. The pointshell has* 85,018 *points total. The common x-axis corresponds to haptic frames. This data was recorded on a three-year-old 3.0 GHz dual Intel Xeon processor with 2.75 Gb of memory, illustrating algorithm's adaptivity to different computer speeds. On the (faster) machine of Table 1, the black timing curve reaches a maximum of ∼800 μs.*

7. Results

Statistics on all haptic rendering examples are provided in Table 1. In our first 6-DoF haptic rendering example, we simulate a detailed deformable bridge in contact with a detailed rigid dinosaur (see Figure 1). We recorded a characteristic manipulandum trajectory, and report simulation data under this trajectory computed offline under different force approximations (see Figure 8). We also present a detailed deformable dragon in contact with a rigid Buddha mesh. Figure 9 demonstrates the benefits of graceful degradation to simulate very large contact areas. Interactive (dis)assembly and path planning are illustrated in Figure 2 by example where the user can manipulate a metallic rigid component in contact with deformable hose geometry typical of modern airplanes. Large hose deformations allow the operator to interactively navigate the component in between the hoses.

Interactive haptic gaming is an emerging application area. In this example, the user can swing a detailed rigid axe to "interact" with an armored rigid knight. This example demonstrates crisp contact with detailed million-point pointshells possible when our method is used for rigid-only simulations.

8. Conclusion

We have demonstrated that real-time haptic rendering of distributed contact between geometrically complex models is possible for both rigid and reduced deformable models. A key ingredient has been the exploitation of low-dimensional parameterizations of deformable models for fast dynamics, point-based deformation bounds, and output-sensitive evaluation of pointshell points and contact normals for time-critical evaluation of contact forces.

Discussion: First, our deformations are not fully general; in the future, fully general complex FEM deformable objects might run at haptics rates on commodity workstations, but this is not the case today. Our deformable model is adaptive: more basis vectors can be added to the basis as processor speeds will increase. While our current demonstrations run on standard Windows or Linux platforms, simulation on operating systems and parallel architectures that can guarantee hard real-time scheduling is a natural area of exploration.

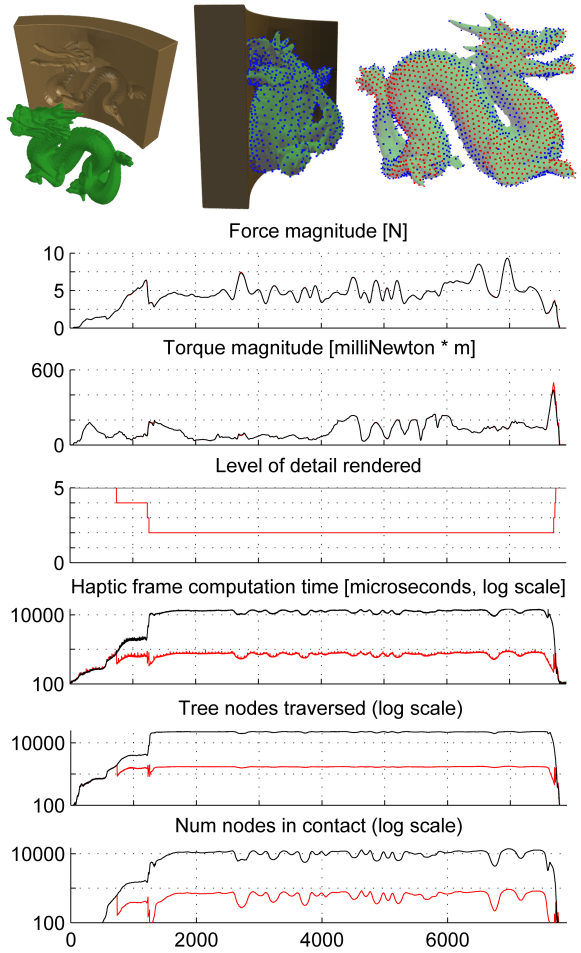

Figure 9: ***Graceful degradation supports large conforming contacts*** *to allow time-critical approximation of very difficult contact configurations (here under 1 millisecond). (**Top Left**) A dragon "peg" inserting into a matching dragon "hole" obtained via CSG difference. (**Top Middle**) dragon manipulated haptically into the hole. (**Top Right**) Large contact area with traversed points in blue (43% of all L2 points), and contacting points in red (21% of all L2 points). (**Bottom Graphs**) Simulation data for a pre-recorded manipulandum trajectory is rendered in two ways: using GD (in red) with warm threshold at 2000 nodes; without GD (in black) with all five tree levels. Same machine as in Table 1. The common x-axis corresponds to haptic frames.*

The simulation can be simplified if the signed distance field is replaced by a simpler implicit function with an analytical formula. Our offset surfaces are not reaching down to the exact geometry in case of polygon "soup" input geometry, when the contact surface is not well-defined. Quality of the offset surface degrades under offsets too small compared to distance field resolution. Large deformations can potentially introduce undersampled pointshell surface regions; however this was not problematic in our case of deformations mostly consisting of large rotations and small strain. Other reduced deformable models should be investigated, including low-resolution models that deform embedded meshes, and multi-resolution models. Much higher rank StVK models are being investigated. Although our implementation is not highly optimized, we did implement some time-critical routines (Uq and ΔNq mtx-vec multiplies) in assembly (SSE2 multimedia instructions), and carefully laid out data in memory to optimize cache performance. A system speedup of about $2\times$ was obtained. Perceptual studies may be useful in determining speed-accuracy trade-offs, such as maximum necessary contact resolution or perception of deformation complexity, similar to other studies done in rigid-rigid contact rendering [OL03]. Another interesting area of future work is simulating friction, which can play an important role in applications such as analysis of mechanical part removability and accessibility. Some applications may be limited by the fact that modification or cutting of the models is prohibited by the preprocessing employed for performance. Although unaddressed in our implementation, large deformations can necessitate self-collision processing, and this remains a significant computational challenge for hard real-time *haptic* rendering applications (c.f. [GKJ*05]).

Acknowledgements: This research was sponsored by the Boeing Company, the National Science Foundation (CAREER-0430528), the Link Foundation, Pixar, Nvidia, Intel, and donation of Maya licenses from Autodesk.

Appendix A: Distance Field Construction

Computing a *signed* distance field is straightforward for closed manifold meshes. In the common case of "polygon soups" we first compute an *unsigned* distance field, then extract an isosurface for some small distance value using marching cubes to obtain a closed manifold mesh [LLVT03] (see also [SOS04]). Next we remove any mesh components completely contained inside another component, and then compute a *signed* distance field with respect to the remaining closed manifold mesh(es). At runtime, we interpolate the distance field grid values to the query location when inside the distance field box. If outside the box, we conservatively estimate $d(p) \geq d_{\text{box}} + d_{\text{boundary}}$, where d_{box} is the shortest distance from the query to the distance field box, and d_{boundary} is the query-independent minimum distance field value on the surface of the distance field box; proof of this inequality follows by connecting the query to its closest point on the surface.

Appendix B: Nested Pointshell Construction

We fit the particles onto a level set of a distance field. If point shell object is given as a closed manifold triangle mesh, we compute a signed distance field and use the zero-distance level set. If it is given as a triangle "soup", we compute an unsigned distance field, and use the level set of some small positive distance value. In either case, call the resulting level set surface $\mathcal{T}$. We also construct a smooth level set *triangulated* surface $\mathcal{S}$ using marching cubes. For unsigned distance fields, we also remove any connected surface components lying completely inside another component.

Our nested hierarchy construction is similar to [Tur01]. We first

"inject" n_1 points of level 1 (coarsest level), by using stratified area-weighted random sampling of $\mathcal{S}$ [Tur92]. We then let the particles repel each other, constrained to $\mathcal{T}$. We do so by alternating particle repulsion and constraint stabilization to $\mathcal{T}$. Our repulsion potential among two particles is radial. Its gradient equals $2(d/R)^3 - 3(d/R)^2 + 1$, where R is the finite influence radius, and d is the Euclidean distance among two particles. Gradient is set to zero for $d \geq R$. At every iteration step, we assemble the repulsion gradients, then move each particle a distance of $-\eta$ in the direction of the net gradient, where $\eta > 0$ is the descent parameter. This step will move the particle (slightly) off the level set surface $\mathcal{T}$. We restore it back to $\mathcal{T}$ by performing a Newton solve using the distance field and its gradients. This two stage process is repeated until particles have sufficiently converged. Then, we permanently fix the position of the particles, and inject n_2 new particles that will form level 2. Again, the level 2 particles are iterated until convergence, when we fix then, and insert level 3 particles. This procedure is repeated until a desired number of levels, or a desired sampling accuracy, has been reached. We use $n_{i+1} = 4n_i$ so that surface sampling resolution about doubles with every level. We also decrease the particle repulsion radius R by a factor of two for each new level. The resulting multi-resolution pointshell has the property that for every level k, the set of points up to level k samples the level set surface $\mathcal{T}$ about uniformly (see Figure 6). Note that the triangle surface $\mathcal{S}$ is only necessary to efficiently draw initial random positions of particles.

Pointshell points need to be assigned (inward-pointing by convention) normals. We do so by locating the closest triangle of $\mathcal{S}$, assigning the normal to be the pseudonormal [BA05] of the closest feature (face, edge, or vertex). Other variations include using barycentrically-weighted pseudonormals, the normalized gradient of the distance field, or, if original geometry is a closed manifold mesh, the pseudonormal of the closest feature on that mesh, which better preserves sharp corners.

References

[AH98] ASTLEY O., HAYWARD V.: Multirate Haptic Simulation Achieved by Coupling Finite Element Meshes Through Norton Equivalents. In *Proc. of the IEEE Int. Conf. on Robotics and Automation* (1998).

[BA05] BÆRENTZEN J., AANÆS H.: Signed distance computation using the angle weighted pseudo-normal. *IEEE Transactions on Visualization and Computer Graphics 11*, 3 (2005), 243–253.

[Bar07] BARBIČ J.: *Real-time Reduced Nonlinear Deformable Objects for Computer Graphics and Haptics*. PhD thesis, Computer Science Department, Carnegie Mellon University, 2007.

[BJ05] BARBIČ J., JAMES D. L.: Real-Time Subspace Integration for St. Venant-Kirchhoff Deformable Models. *ACM Transactions on Graphics (SIGGRAPH 2005) 24*, 3 (Aug. 2005).

[BNC96] BRO-NIELSEN M., COTIN S.: Real-time Volumetric Deformable Models for Surgery Simulation using Finite Elements and Condensation. *Comp. Graphics Forum 15*, 3 (1996), 57–66.

[CDA99] COTIN S., DELINGETTE H., AYACHE N.: Realtime Elastic Deformations of Soft Tissues for Surgery Simulation. *IEEE Trans. on Vis. and Comp. Graphics 5*, 1 (1999), 62–73.

[CGSS93] COLGATE J., GRAFING P., STANLEY M., SCHENKEL G.: Implementation of Stiff Virtual Walls in Force-Reflecting Interfaces. In *Proc. of IEEE Virtual Reality Annual Int. Symp.* (1993), IEEE, pp. 202–208.

[DDCB01] DEBUNNE G., DESBRUN M., CANI M.-P., BARR A. H.: Dynamic Real-Time Deformations Using Space & Time Adaptive Sampling. In *Proc. of ACM SIGGRAPH 2001* (August 2001), pp. 31–36.

[DDKA06] DURIEZ C., DUBOIS F., KHEDDAR A., ANDRIOT C.: Realistic Haptic Rendering of Interacting Deformable Objects in Virtual Environments. *IEEE Trans. on Vis. and Comp. Graphics 12*, 1 (2006), 36–47.

[GKJ*05] GOVINDARAJU N. K., KNOTT D., JAIN N., KABUL I., TAMSTORF R., GAYLE R., LIN M. C., MANOCHA D.: Interactive collision detection between deformable models using chromatic decomposition. *ACM Transactions on Graphics 24*, 3 (Aug. 2005), 991–999.

[Hub95] HUBBARD P. M.: *Collision Detection for Interactive Graphics Applications*. PhD thesis, Department of Comp. Science, Brown University, 1995.

[JCC06] JUN S., CHOI J., CHO M.: Physics-based s-adaptive haptic simulation for deformable object. In *Proc. of the Symp. on Haptic Interfaces For Virtual Environment and Teleoperator Systems* (2006), pp. 72–78.

[JP01] JAMES D. L., PAI D. K.: A Unified Treatment of Elastostatic Contact Simulation for Real Time Haptics. *Haptics-e, The Electronic J. of Haptics Research (www.haptics-e.org) 2*, 1 (September 2001).

[JP02] JAMES D. L., PAI D. K.: DyRT: Dynamic Response Textures for Real Time Deformation Simulation With Graphics Hardware. *ACM Trans. on Graphics 21*, 3 (July 2002), 582–585.

[JP03] JAMES D. L., PAI D. K.: Multiresolution Green's Function Methods for Interactive Simulation of Large-scale Elastostatic Objects. *ACM Trans. on Graphics 22*, 1 (2003), 47–82.

[JP04] JAMES D. L., PAI D. K.: BD-Tree: Output-Sensitive Collision Detection for Reduced Deformable Models. *ACM Trans. on Graphics 23*, 3 (2004), 393–398.

[KcM00] KÜHNAPFEL U., ÇAKMAK H., MAASSH.: Endoscopic surgery training using virtual reality and deformable tissue simulation. *Computers & Graphics 24* (2000), 671–682.

[KMH*04] KEISER R., MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Contact Handling for Deformable Point-Based Objects. In *Proc. of Vision, Modeling, Visualization 2004* (2004), pp. 339–346.

[KOLM03] KIM Y. J., OTADUY M. A., LIN M. C., MANOCHA D.: Six degree-of-freedom haptic display using incremental and localized computations. *Presence-Teleoperators and Virtual Environments 12*, 3 (2003), 277–295.

[LD07] LAYCOCK S., DAY A.: A Survey of Haptic Rendering Techniques. *Computer Graphics Forum 26* (2007), 50–65.

[LLVT03] LEWINER T., LOPES H., VIEIRA A. W., TAVARES G.: Efficient implementation of Marching Cubes' cases with topological guarantees. *Journal of Graphics Tools 8*, 2 (2003), 1–15.

[MO06] MENDOZA C., O'SULLIVAN C.: Interruptible collision detection for deformable objects. *Computer & Graphics 30*, 2 (2006).

[MPT99] MCNEELY W. A., PUTERBAUGH K. D., TROY J. J.: Six degree-of-freedom haptic rendering using voxel sampling. In *Proc. of ACM SIGGRAPH 99* (1999), ACM, pp. 401–408.

[MPT06] MCNEELY W., PUTERBAUGH K., TROY J.: Voxel-Based 6-DOF Haptic Rendering Improvements. *Haptics-e 3*, 7 (2006).

[OG07] OTADUY M. A., GROSS M.: Transparent Rendering of Tool Contact with Compliant Environments. In *Proc. of the World Haptics Conference* (2007), pp. 225–230.

[OL03] OTADUY M. A., LIN M. C.: Sensation Preserving Simplification for Haptic Rendering. In *Proc. of ACM SIGGRAPH 2003* (2003), ACM, pp. 543–553.

[OL06] OTADUY M. A., LIN M. C.: A Modular Haptic Rendering Algorithm for Stable and Transparent 6-DoF Manipulation. *IEEE Trans. on Robotics 22*, 4 (2006), 751–762.

[PPG04] PAULY M., PAI D. K., GUIBAS L.: Quasi-Rigid Objects in Contact. In *Proc. of the Symp. on Comp. Animation 2004* (2004).

[RKK97] RUSPINI D. C., KOLAROV K., KHATIB O.: The Haptic Display of Complex Graphical Environments. In *Proc. of ACM SIGGRAPH 97* (Aug. 1997), Whitted T., (Ed.), Addison Wesley, pp. 345–352.

[RL00] RUSINKIEWICZ S., LEVOY M.: Qsplat: A multiresolution point rendering system for large meshes. In *Proceedings of ACM SIGGRAPH 2000* (July 2000), Computer Graphics Proceedings, Annual Conference Series, pp. 343–352.

[RPP*01] RENZ M., PREUSCHE C., PÖTKE M., KRIEGEL H.-P., HIRZINGER G.: Stable haptic interaction with virtual environments using an adapted voxmap-pointshell algorithm. In *Proc. of Eurohaptics* (2001), pp. 149–154.

[SOS04] SHEN C., O'BRIEN J. F., SHEWCHUK J. R.: Interpolating and approximating implicit surfaces from polygon soup. In *Proceedings of ACM SIGGRAPH 2004* (Aug. 2004), ACM Press, pp. 896–904.

[Tur92] TURK G.: Re-Tiling Polygonal Surfaces. *Computer Graphics (Proc. of ACM SIGGRAPH 92) 26*, 2 (1992), 55–64.

[Tur01] TURK G.: Texture Synthesis on Surfaces. In *Proceedings of ACM SIGGRAPH 2001* (Aug. 2001), Computer Graphics Proceedings, Annual Conference Series, pp. 347–354.

[WH94] WITKIN A. P., HECKBERT P. S.: Using particles to sample and control implicit surfaces. In *Proceedings of SIGGRAPH 94* (July 1994), Computer Graphics Proceedings, Annual Conference Series, pp. 269–278.

[WM03] WAN M., MCNEELY W. A.: Quasi-Static Approximation for 6 Degrees-of-Freedom Haptic Rendering. In *Proc. of IEEE Visualization 2003* (2003), pp. 257–262.

[ZS95] ZILLES C., SALISBURY J.: A Constraint-based God-object Method for Haptics Display. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (1995), IEEE, pp. 146–151.