Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2006)
M.-P. Cani, J. O'Brien (Editors)

Fast Simulation of Deformable Models in Contact Using Dynamic Deformation Textures

Nico Galoppo[+] Miguel A. Otaduy[⋆] Paul Mecklenburg[+] Markus Gross[⋆] Ming C. Lin[+]

[+]{nico,prm,lin}@cs.unc.edu, UNC Chapel Hill [⋆]{otaduy,grossm}@inf.ethz.ch, ETH Zurich

Abstract

We present an efficient algorithm for simulating contacts between deformable bodies with high-resolution surface geometry using dynamic deformation textures*, which reformulate the 3D elastoplastic deformation and collision handling on a 2D parametric atlas to reduce the extremely high number of degrees of freedom in such a computationally demanding simulation. We perform proximity queries for deformable bodies using a two-stage algorithm directly on dynamic deformation textures, resulting in output-sensitive collision detection that is independent of the combinatorial complexity of the deforming meshes. We present a robust, parallelizable formulation for computing constraint forces using implicit methods that exploits the structure of the motion equations to achieve highly stable simulation, while taking large time steps with inhomogeneous materials. The dynamic deformation textures can also be used directly for real-time shading and can easily be implemented using SIMD architecture on commodity hardware. We show that our approach, complementing existing pioneering work, offers significant computational advantages on challenging contact scenarios in dynamic simulation of deformable bodies.*

1. Introduction

Modeling and simulation of deformable objects has received much recent attention in computer graphics: for virtual characters, animals, faces, hair, soft tissue, cloth, etc. Among the techniques for simulating deformable bodies, many have focused on computing global deformations in a stable, accurate, and efficient manner [TW88, MG04, TSIF05, BJ05]; and some have aimed at efficiently simulating the effects of localized contacts [BNC96, JP99]. However, most of the existing methods do not explicitly address the problem of simulating deformation due to large contact areas with high-resolution surface geometry. Due to the inherently numerous degrees of freedom (DoFs) involved in this type of deformation, the complexity of the problem tends to be intractable for real-time computations.

Main Results: In this paper, we present a novel and fast simulation framework for deformable bodies in contact, based on the projection of the 3-dimensional deformation field onto a lower-dimensional space to efficiently deal with the many DoFs arising from large contact regions and high-resolution geometry. We assume that a simulated object can be modeled as a rigid (and possibly articulated) core covered by a layer of deformable material, and that the deformation field of the surface can be expressed as a function in two-dimensional, parametric atlases, called *dynamic deformation textures*. Layered representations have been proposed before for simulating skeletal deformations [CHP89, Gas98, CBC*05], as many real-world solids retain their core structures under large deformations. Examples include animated characters, human bodies, furniture, toys, footballs, tires, etc. We derive an implicit yet highly parallelizable solution to dynamic deformations using linear elasticity theory (with separation of rigid motion), continuum Lagrangian mechanics, FEM discretization, and constraint-based contact response with Lagrange multipliers. Our approach offers the following advantages:

- With the reformulation of the 3-dimensional elastoplastic deformations and collision processing on 2-dimensional dynamic deformation textures, the resulting system achieves fast and robust simulations of contacts between deformable bodies with rich, high-resolution surface geometry.
- Using a two-stage collision detection algorithm, our proximity queries are scalable and output-sensitive, i.e. the performance of the queries does not directly depend on the complexity of the surface meshes.
- By decoupling the parallel update of surface displacements from the update of the core DoFs, our efficient implicit formulation enables fast, stable simulations of heterogeneous materials under large time steps.
- Our constraint-based collision response, using Lagrange multipliers and approximate implicit integration of elas-

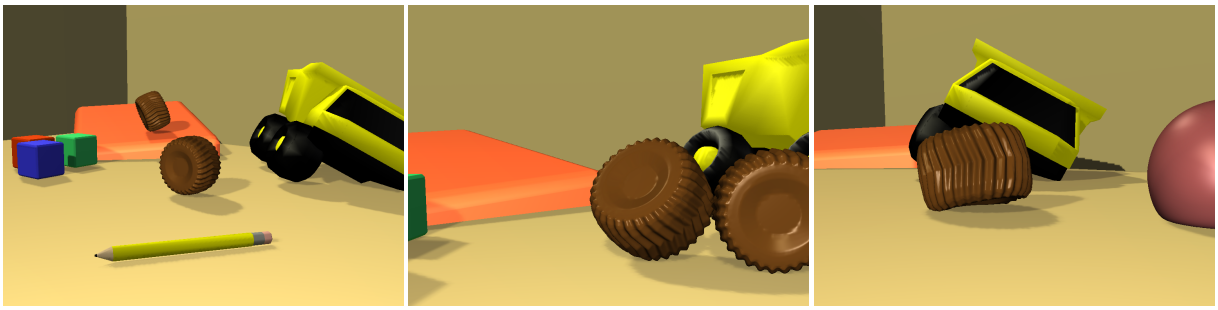

Figure 1: ***Deformable Objects Roll and Collide in the Playground.***

tic forces, provides fast and responsive contact handling, alleviating time-step restrictions of previous impulsive methods.

Our mathematical formulation of dynamic simulation and contact processing, along with the use of dynamic deformation textures, is especially well suited for realization on commodity SIMD or parallel architectures, such as graphics processing units (GPU), Cell processors, and physics processing units (PPU). The surface detail attributes stored in dynamic deformation textures can also be used directly for high-quality real-time shaders. We have demonstrated the effectiveness of dynamic deformation textures for fast simulation of deformable objects in contact on various challenging scenarios. Without any precomputation of dynamics or significant storage requirement, we are able to simulate large surface deformations with detailed surface geometry at average simulation rates of $1-6$ fps, on a 3.4 GHz Pentium 4 PC with NVidia GeForce 7800.

Organization: The rest of the paper is structured as follows. Section 2 discusses the related work on simulation of deformable models. We introduce our novel representation and give an overview of our approach in Section 3. Section 4 describes the formulation and solution of motion equations. Section 5 presents our collision detection and contact handling techniques. Section 6 summarizes the simulation algorithm, and explains its parallel implementation on graphics hardware. Finally, we show results in Section 7 and discuss possible future research directions in Section 8.

2. Related Work

Physically-based simulation of deformable bodies has been widely studied in computer graphics during the last two decades [TPBF87]. For an extended discussion on related work, we refer the readers to recent summaries on the simulation of deformation [NMK*05] and collision detection [TKH*05].

To date, finite element methods (FEM) have often been used to discretize the partial differential equations that describe the behavior of dynamic continuum deformation models, and result in (generally nonlinear) second-order ordinary differential motion equations [Sha89]. Non-linear deformation models together with explicit integration enable fast simulation of soft deformable models of moderate complexity [ZC99]. On the other hand, linear deformation models lead to linear motion equations that enable fast implicit integration or quasi-static approximation with large time steps and heterogeneous materials. Our formulation of the motion equations for layered deformable objects is kindred to the seminal work by Terzopoulos and Witkin [TW88] that handles large deformations with linear deformation models by extracting a rigid body motion from the deformation field. This was later extended to account for a deformable reference shape and constraints [MT92]. Whereas Terzopoulos and Witkin [TW88] suggested an explicit integration of rigid motion, we derive a fully implicit integration that ensures stable, two-way elastic coupling with large simulation time steps. Corotational techniques [MG04] exploit linear deformation models and estimate a local reference frame at each mesh element. However, they do not consider centripetal and Coriolis forces introduced by the moving reference frame.

Several recent techniques have been proposed to reduce the number of degrees of freedom (DoFs) in deformation simulations. For example, multiresolution methods [DDCB01, GKS02, CGC*02] concentrate computations with many DoFs at locations where high accuracy or high detail is required. Reduced coordinate methods based on modal analysis [JP02, HSO03, BJ05] achieve rich deformations with a low computational cost by describing global deformations as the combination of a few DoFs. Existing multiresolution and reduced coordinate methods implicitly assume that a small number of DoFs or a few global deformation bases are sufficient to describe meaningful and possibly very large deformations. Complementing the existing techniques, our approach is designed to efficiently simulate colliding deformable bodies with high-resolution geometry over large contact areas.

Our proposed model is akin to other methods that focus on the surface of the deformable model. Condensation [BNC96] and boundary element methods (BEM) [JP99] exploit linear elasticity to solve a system that is dense w.r.t. the number of surface nodes. James and Pai [JP99] further optimized this approach by performing incremental updates in situations with contact coherence. In contrast, our proposed model requires the solution of linear systems that

are *sparse* w.r.t. the number of surface nodes. Layered deformable models [CHP89, Gas98, CBC*05] overlay layers of deformable material on top of an articulated skeleton that drives the motion. Upon contact, these models typically produce only surface deformations and are often not designed to capture the two-way coupling of the global motion of the colliding objects. Novel rigid body models with compliance [PPG04] can be regarded as a specific type of layered deformable model, designed to alleviate singularities in contact computation for rigid bodies. Similar to our approach, they capture the two-way coupling of global motion, but we employ more accurate deformation models that enable the simulation of larger deformations. Texture-based representations have been used for animating surface deformations excited by global deformation modes [JP02] or as the result of contact [SOH99, WRM05]. Contrary to previous texture-based techniques, our approach captures the coupling between global motion, contact forces, and deformations.

Collision response is typically applied in one of two ways: using penalty methods or enforcing constraints. Penalty methods [WVS90, HFS03] are easy to formulate, but they rely on the computation of penetration depth or distance tolerance to produce collision response. Interpenetrations and tolerances may be alleviated by using stiff penalty forces along with implicit integration methods, but this approach results in a coupling of the motion equations of colliding bodies, and penalty methods lose their original simplicity. Formulation of contact constraints based on the Signorini problem leads to a linear complementarity problem (LCP) on contact forces and displacements [DAK04, PPG04]. For accelerating the solution to the LCP, Pauly et al. [PPG04] suggest a hierarchical representation that smooths the deformations.

Our contact handling approach belongs to a family of methods that formulate contact constraints and apply contact impulses on each colliding object independently [ZC99, BFA02, CW05]. Previous methods apply impulses explicitly and, due to the Courant condition, require small time steps to propagate the effect of collision response through the objects [PPG04]. We propose a novel efficient and highly parallelizable solution, which enables the simulation to take very large time steps with robust contact handling, based on Lagrange multipliers, implicit integration, and a plausible approximation of elastic deformation forces.

3. Overview of the Representation

In this section, we present our layered representation of deformable objects. We discuss the 2-dimensional parameterization of the deformation field, its linear FEM discretization, and texture-based storage. We also introduce the generalized set of decoupled rigid and deformation coordinates.

3.1. Parameterization of Layered Deformable Objects

We model each deformable object as a *core* covered by a layer of (possibly heterogeneous) deformable material. We restrict the formulation to a rigid core, but it could be extended to articulated bodies. The layered representation enables modeling (i) both large and small scale deformations over large regions of the object's surface, (ii) global deformations of skeletal nature, and (iii) two-way dynamic coupling between the global motion of the object and the surface deformations produced during contact.

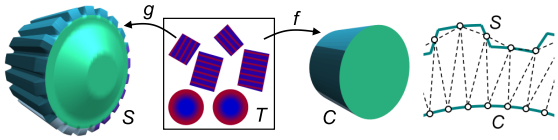

Figure 2: ***Deformable Object Representation.*** *Deformable surface S (52K triangles) and core C (252 triangles) of a gear, showing its parameterized atlas T. The color coding reflects the mapping $g \circ f^{-1} : C \rightarrow S$. The dynamic deformation texture T (256×256) stores the displacement field values on the surface. On the right, 2D schematic figure showing a slice of the tetrahedral meshing. The gear contains 28K simulation nodes on the surface and 161K tetrahedra, allowing the simulation of highly detailed deformations.*

A body frame is attached to the core, with rotation $\mathbf{R}$ and position of the center of mass $\mathbf{c}$. The world position $\mathbf{x}$ of a point in the object can be expressed in terms of its body-frame position $\mathbf{u}$ as $\mathbf{x} = \mathbf{c} + \mathbf{R}\mathbf{u}$. In the deformable layer, $\mathbf{u}$ can be decomposed into a constant undeformed component $\mathbf{u}_o$ and a displacement $\mathbf{u}_e$, hence $\mathbf{u} = \mathbf{u}_o + \mathbf{u}_e$.

We represent 3-dimensional deformations in a 2-dimensional parametric domain. For that purpose, we parameterize the surfaces of the core and the deformable object. A surface patch $C \subset \mathbb{R}^3$ of the core can be defined by a mapping f from a domain $T \subset \mathbb{R}^2$, $f : T \rightarrow C$. We enforce a one-to-one correspondence between points on the surface of the deformable object and points on the surface of the core. Then a corresponding surface patch $S \subset \mathbb{R}^3$ of the deformable object can be defined by a mapping $g : T \rightarrow S$, and the correspondence by $g \circ f^{-1}$. Based on the mapping g, the body-frame position $\mathbf{u}$ of the surface of the object and the displacement field $\mathbf{u}_e$ can be expressed as 2-dimensional functions $\mathbf{u}(s,t)$ and $\mathbf{u}_e(s,t)$.

The one-to-one correspondence can be achieved by appropriately modeling the core. One option is to parameterize the surface S in its rest position and obtain the core C by surface decimation while preserving the parameterization [COM98, SSGH01]. Another option is to parameterize the core C and model the surface S by successive subdivision and addition of geometric detail [ZSS97]. The surfaces of the core and of the deformable object may be partitioned into multiple patches, and each patch parameterized independently.

3.2. Discretization and Generalized Coordinates

A regular sampling of the planar domain T can be regarded as a texture atlas, which we refer to as *dynamic deforma-*

tion texture. We also refer to grid points (s,t) as texels. Each texel $(s,t) \in T$ maps to two corresponding points $f(s,t)$ and $g(s,t)$ on the surfaces of the core and the deformable object. The regular sampling of T and the correspondence of surface points define implicitly a meshing with a single layer of tetrahedral elements, as shown in Figure 2. By applying classical approximation methods such as FEM, the continuous displacement field $\mathbf{u}_e$ on the deformable layer can be approximated from the values at a finite set of nodes. Since $\mathbf{u}_e = 0$ at points on the core, $\mathbf{u}_e$ can be approximated entirely from the values $\mathbf{u}_e(s,t)$ at surface nodes. Effectively, each texel $(s,t) \in T$ maps to a simulation node $g(s,t)$ in the FEM discretization. Simulation variables defined per-node, such as velocities, forces, mass and stiffness values, etc. can also be stored in texture atlases. Note that the implicitly defined texture-based meshing is not consistent at patch boundaries, which require special yet simple treatment as discussed in Section 6.

The displacements of the surface nodes are packed in a vector of *elastic coordinates* $\mathbf{q}_e \in \mathbb{R}^n$. Together with the *core coordinates* $\mathbf{q}_c = \begin{bmatrix} \mathbf{c} \\ \theta \end{bmatrix} \in \mathbb{R}^7$, they form the generalized coordinates $\mathbf{q} = \begin{bmatrix} \mathbf{q}_c \\ \mathbf{q}_e \end{bmatrix}$ that describe the state of a layered deformable object. We choose quaternions to represent the orientation θ. Given a (position-dependent) shape matrix $\mathbf{S}$, the displacement of a point in the deformable layer can be expressed in compact matrix form as $\mathbf{u}_e = \mathbf{S}\mathbf{q}_e$. Then the world-frame position of a material point can be written as $\mathbf{x} = \mathbf{c} + \mathbf{R}(\mathbf{u}_o + \mathbf{S}\mathbf{q}_e)$. With linear basis functions, $\mathbf{S}$ is linear in the barycentric coordinates for each mesh element [BNC96].

Given ω, the angular velocity of the core expressed in body-frame coordinates, we define a velocity state vector $\mathbf{v} = \begin{bmatrix} \mathbf{v}_c \\ \mathbf{v}_e \end{bmatrix}$, where $\mathbf{v}_c = \begin{bmatrix} \dot{\mathbf{c}} \\ \omega \end{bmatrix}$ and $\mathbf{v}_e = \dot{\mathbf{q}}_e$. As shown in Appendix A, the velocity state vector and the generalized coordinates are related by $\mathbf{v} = \mathbf{P}\dot{\mathbf{q}}$ and $\dot{\mathbf{q}} = \mathbf{P}^+\mathbf{v}$, where $\mathbf{P}$ and $\mathbf{P}^+$ are matrices that encapsulate the relation between ω and the derivative of a quaternion.

4. Dynamic Deformations

In this section, we first discuss the formulation of the motion equations from Lagrangian mechanics, linear elasticity theory, and linear FEM discretization. Then we describe an efficient parallelizable formulation of implicit integration that exploits our representation based on dynamic deformation textures by separating the update of core and elastic velocities, but maintaining responsive two-way coupling between the core and the deformable layer.

4.1. Equations of Motion

The motion equations of a deformable body can be derived from Lagrangian continuum mechanics [GPS02, Sha89]. Using linear elasticity theory and linear FEM, and by formulating the displacement field in the floating frame of reference, elastic forces are linear w.r.t. the elastic coordinates $\mathbf{q}_e$ and invariant under rigid motion of the core [Sha89, TW88]. We denote mass, damping, and stiffness matrices as $\mathbf{M}$, $\mathbf{D}$, and $\mathbf{K}$, generalized external forces as $\mathbf{Q}$, and a quadratic velocity vector that captures the inertial effects of centripetal and Coriolis forces as $\mathbf{Q}_v$. We obtain the ordinary differential motion equations:

Figure 3: ***Simulation of Heterogeneous Materials.*** *Our efficient decoupled implicit integration enables fast simulation of a heterogeneous cylinder.*

$$\begin{cases} \mathbf{M}\dot{\mathbf{v}} = \mathbf{Q} + \mathbf{Q}_v - \mathbf{K}\mathbf{q} - \mathbf{D}\mathbf{v} = \mathbf{F}, \\ \dot{\mathbf{q}} = \mathbf{P}^+\mathbf{v}. \end{cases} \tag{1}$$

(A detailed description of the derivation can be found in [Sha89].) We choose to model $\mathbf{D}$ as a diagonal matrix acting only on the elastic coordinates. The mass and stiffness matrices, on the other hand, present the following structure:

$$\mathbf{M} = \begin{pmatrix} \mathbf{M}_c & \mathbf{M}_{ce} \\ \mathbf{M}_{ce}^T & \mathbf{M}_e \end{pmatrix} \quad \text{and} \quad \mathbf{K} = \begin{pmatrix} 0 & 0 \\ 0 & \mathbf{K}_e \end{pmatrix}. \tag{2}$$

$\mathbf{M}_c \in \mathbb{R}^{6\times 6}$ and $\mathbf{M}_{ce} \in \mathbb{R}^{6\times n}$ are dense, and we approximate $\mathbf{M}_e \in \mathbb{R}^{n\times n}$ with a diagonal form by applying mass lumping. As shown by the structure of $\mathbf{K}$, the elastic forces depend only on the elastic coordinates $\mathbf{q}_e$. However, these affect the core coordinates as well, by inertial coupling through $\mathbf{M}_{ce}$. The submatrix $\mathbf{K}_e \in \mathbb{R}^{n\times n}$ is constant and sparse, with at most 21 non-zero terms per row, due to the regular texture-based meshing.

4.2. Efficient Decoupled Implicit Integration

We have opted for an implicit backward Euler discretization of the motion equations, enabling simulation of heterogeneous materials without letting the time step be governed by stiff regions, as shown in Fig. 3. Backward Euler discretization yields a nonlinear set of equations, which can be linearized using a first-order approximation of the force $\mathbf{F}(t+\Delta t)$ w.r.t. the state vectors $\mathbf{v}$ and $\mathbf{q}$. After algebraic manipulation, and assuming a constant mass matrix $\mathbf{M}$ during each time step, we obtain the equation for the velocity update $\Delta\mathbf{v}$:

$$\left(\mathbf{M} - \Delta t\frac{\partial \mathbf{F}}{\partial \mathbf{v}} - \Delta t^2\frac{\partial \mathbf{F}}{\partial \mathbf{q}}\mathbf{P}^+\right)\Delta\mathbf{v} = \Delta t\mathbf{F}(t) + \Delta t^2\frac{\partial \mathbf{F}}{\partial \mathbf{q}}\mathbf{P}^+\mathbf{v}(t). \tag{3}$$

We define the *implicit mass matrix* $\tilde{\mathbf{M}}$ and *implicit force vector* $\tilde{\mathbf{F}}$ by gathering terms in (3):

$$\tilde{\mathbf{M}}\Delta\mathbf{v} = \Delta t\tilde{\mathbf{F}}(t). \tag{4}$$

Due to the linearity of (4), we can first compute a collision-free velocity update, and then add the effect of contact forces (Section 5.2) to produce the constrained velocity update.

The implicit mass matrix $\tilde{\mathbf{M}}$ does not lend to an efficient solution of the linear system (4). Instead, we propose an efficient solution that exploits the block structure of $\tilde{\mathbf{M}}$ and $\tilde{\mathbf{F}}$ in (2). We derive the following equations to decouple the update of core velocities $\mathbf{v}_c$ and elastic velocities $\mathbf{v}_e$:

$$\Delta\mathbf{v}_c = \tilde{\mathbf{M}}_{\text{cond}}^{-1}(\Delta t\tilde{\mathbf{F}}_c - \Delta t\tilde{\mathbf{M}}_{ce}\tilde{\mathbf{M}}_e^{-1}\tilde{\mathbf{F}}_e), \quad (5)$$

$$\Delta\mathbf{v}_e = \Delta t\tilde{\mathbf{M}}_e^{-1}\tilde{\mathbf{F}}_e - \tilde{\mathbf{M}}_e^{-1}\tilde{\mathbf{M}}_{ec}\Delta\mathbf{v}_c, \quad (6)$$

in which the condensed matrix $\tilde{\mathbf{M}}_{\text{cond}} = \tilde{\mathbf{M}}_c - \tilde{\mathbf{M}}_{ce}\tilde{\mathbf{M}}_e^{-1}\tilde{\mathbf{M}}_{ec} \in \mathbb{R}^{6\times 6}$. This differs from other surface-oriented approaches [BNC96], where the size of the condensed matrix is governed by the number of surface nodes. The advantage of the decoupling lies in the structure of the systems to be solved. First, we solve two linear systems $\tilde{\mathbf{M}}_e\mathbf{y} = \tilde{\mathbf{F}}_e$ and $\tilde{\mathbf{M}}_e\mathbf{Y} = \tilde{\mathbf{M}}_{ec}$. Then we update $\mathbf{v}_c$ by solving the condensed system (5). Finally, the elastic velocities $\mathbf{v}_e$ can be solved through (6) in a highly parallel manner.

The two $n \times n$ linear systems to be solved imply the matrix $\tilde{\mathbf{M}}_e$ which, omitting external forces, can be written as:

$$\tilde{\mathbf{M}}_e = \mathbf{M}_e - \Delta t\frac{\partial\mathbf{Q}_{ve}}{\partial\mathbf{v}_e} - \Delta t^2\frac{\partial\mathbf{Q}_{ve}}{\partial\mathbf{q}_e} + \Delta t\mathbf{D}_e + \Delta t^2\mathbf{K}_e. \quad (7)$$

We approximate the Jacobians of the quadratic velocity vector $\mathbf{Q}_{ve}$ by their diagonal forms. In this way, the matrix $\tilde{\mathbf{M}}_e$ is sparse, symmetric and positive definite; thus it can be solved efficiently using iterative methods. Moreover, as dicussed in Section 6, the regularity of the matrix $\mathbf{K}_e$, due to our texture-based representation, enables a highly parallelizable implementation of the iterative solvers.

5. Collision Detection and Response

We start this section with a description of our texture-based collision detection algorithm for deformable surfaces with high-resolution geometry. We first describe the image-space detection of contact constraints and how these are mapped to the texture-based simulation domain. Then we describe the formulation of velocity constraints in the generalized coordinate setting using Lagrange multipliers. For enhanced two-way dynamic coupling between core and deformable layer under collisions, we propose a solution of collision response based on the implicit integration of constraint forces, and we present an efficient numerical solution.

5.1. Texture-Based Collision Detection

We propose to perform collision detection between two deformable objects A and B in two steps: (i) identify contact regions with object-space techniques using low-resolution models of the objects; (ii) compute contact information in the contact regions using image-space techniques and high-resolution displacement fields. A similar approach has been exploited for estimating the penetration depth value between rigid, high-resolution objects [OJSL04], whereas we perform collision handling of deformable objects and compute contact information for many colliding surface points.

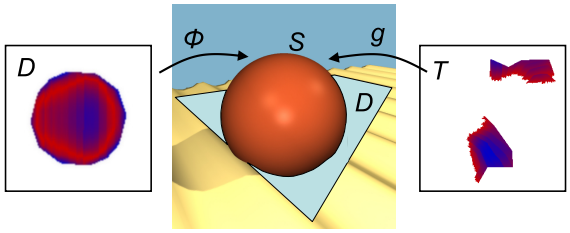

Figure 4: ***Texture-Based Collision Detection Process.*** *Center: A sphere S collides with a textured terrain. Left: Contact plane D for texture-based collision detection, and mapping $\phi : D \rightarrow S$. The contact plane shows the penetration depth. Right: Dynamic deformation texture T, and mapping $g : T \rightarrow S$. The penetration depth is projected from D to T, and is available for collision response.*

In the object-space collision detection step, we identify patches of the core surfaces closer than a distance tolerance that bounds the high-resolution deformable surfaces. We employ existing acceleration methods based on convex hull hierarchies [EL01]. Given a contact region between core surface patches $C_A \subset \mathbb{R}^3$ and $C_B \subset \mathbb{R}^3$, we identify a contact plane $D \subset \mathbb{R}^2$ as the plane passing between the contact points and oriented according to the contact normal. By orthonormal projection of C_A (and similarly for C_B) onto D, we define a mapping $h_A : D \rightarrow C_A$. Due to the one-to-one correspondence between patches on the surface core and patches on the deformable surface, a contact between core surface patches C_A and C_B defines a potential contact between corresponding deformable surface patches S_A and S_B.

Given a patch S on a high-resolution deformable surface, and the mappings g and f defined in Section 3.1, $\phi = g \circ f^{-1} \circ h$ defines a mapping $\phi : D \rightarrow S$ from the contact plane D to the patch S, through the core patch C and the texture atlas T, as shown in Figure 4. Similarly to the sampling of the texture atlas T, we regularly sample the contact plane D. Then, each texel $(u,v) \in D$ maps to a point $\phi(u,v)$ on the high-resolution patch S.

For each texel $(u,v) \in D$, we perform high-resolution collision detection by testing the distance between points $\phi_A(u,v) \in S_A$ and $\phi_B(u,v) \in S_B$ along the normal direction of D. If the points are penetrating, we identify a contact constraint and we compute the contact normal $\mathbf{n}$ as the average surface normal. We also approximate the penetration depth as $d = \mathbf{n}^T(\phi_B(u,v) - \phi_A(u,v))$ for applying constraint correction. This approximation is affected by texture distortion, but we have not found noticeable errors in our examples.

Contact constraint information can be transferred to a texture atlas T via the mapping $f^{-1} \circ h$ and made readily available for the computation of collision response at the simula-

tion nodes, as shown in Figure 4. For accuracy of collision detection, it is convenient to sample the contact plane D at a higher density than the texture atlas T. As a result, multiple colliding points $(u,v) \in D$ may map to the same simulation node $(s,t) \in T$. In such cases, we keep only the constraint information from the colliding point with the largest penetration depth value.

5.2. Contact Resolution

Colliding surface nodes are prevented from penetrating other objects by the application of contact constraint forces. We first describe the handling of fixed, frictionless constraints, and then we extend the algorithm to moving constraints with friction. We define pre-impact velocities $\mathbf{v}^-$ computed by solving (4), post-impact velocities $\mathbf{v}^+$ and collision impulse $\delta\mathbf{v} = \mathbf{v}^+ - \mathbf{v}^-$. The contact constraints are expressed in the world-frame velocities of the colliding nodes, and must be transformed to the generalized coordinate setting by (19) in Appendix A. A planar constraint $\mathbf{n}$ acting at a node i produces an elastic collision impulse with coefficient of restitution ε governed by:

$$\mathbf{n}^T \left(\dot{\mathbf{x}}_i^- + \frac{\delta \dot{\mathbf{x}}_i}{1+\varepsilon} \right) = \mathbf{j} \left(\mathbf{v}^- + \frac{\delta \mathbf{v}}{1+\varepsilon} \right) = 0. \tag{8}$$

The vector $\mathbf{j} = \mathbf{n}^T \mathbf{L}_i = \begin{bmatrix} \mathbf{n}^T & -\mathbf{n}^T \mathbf{R} \tilde{\mathbf{u}}_i & \mathbf{n}^T \mathbf{R} \mathbf{S}_i \end{bmatrix}$ represents the generalized constraint normal, where $\mathbf{S}_i$ indicates the position-dependent matrix $\mathbf{S}$ evaluated at node i. Note that $\mathbf{S}_i \mathbf{v}_e$ selects the i^{th} block component from $\mathbf{v}_e$. The velocity constraints can be jointly formulated with a generalized constraint matrix $\mathbf{J} \in \mathbb{R}^{m \times (6+n)}$, where m is the number of colliding surface nodes:

$$\mathbf{J} \left(\mathbf{v}^- + \frac{\delta \mathbf{v}}{1+\varepsilon} \right) = 0. \tag{9}$$

In order to compute the collision impulse, we add a constraint force vector $\mathbf{J}^T \lambda$ to the external forces $\mathbf{Q}$ in (1). Here, λ is a vector of Lagrange multipliers. Typically, the collision impulse is solved by explicitly integrating the constraint forces, which is equivalent to applying an instantaneous change of momentum to the surface nodes at the end of each time step. Unfortunately, this approach requires small simulation time steps for the correct propagation of pressure waves induced by collision response [PPG04]. With explicit integration and large time steps, the elastic deformation forces cannot counteract the momentum of the core upon collision, and the core may penetrate the constraints.

We propose the computation of the collision impulse through implicit integration which produces a robust and responsive reaction of the core with large time steps. Due to linearity of (4) w.r.t. the vector of forces $\tilde{\mathbf{F}}$, we can compute the collision impulse separately by solving:

$$\tilde{\mathbf{M}} \delta \mathbf{v} = \Delta t \mathbf{J}^T \lambda. \tag{10}$$

From (9) and (10), we obtain:

$$\mathbf{J} \tilde{\mathbf{M}}^{-1} \mathbf{J}^T \lambda = -\frac{1+\varepsilon}{\Delta t} \mathbf{J} \mathbf{v}^-. \tag{11}$$

After solving this equation for λ, we can compute the constraint forces $\mathbf{J}^T \lambda$ and perform the efficient decoupled implicit velocity update described in Section 4.2 to compute the post-impact velocities $\mathbf{v}^+$.

5.3. Efficient Decoupled Contact Resolution

The matrix $\mathbf{J} \tilde{\mathbf{M}}^{-1} \mathbf{J}^T$ is dense, and the computation of λ through (11) is computationally expensive. Instead, we propose to decouple (11) by exploiting the structure of $\mathbf{J} = \begin{bmatrix} \mathbf{J}_c & \mathbf{J}_e \end{bmatrix}$, which can easily be derived from the individual node velocity constraints (8). $\mathbf{J}_c \in \mathbb{R}^{m \times 6}$ is dense, and $\mathbf{J}_e \in \mathbb{R}^{m \times n}$ presents one non-zero 1×3 block per row. Equation (11) can be rewritten as:

$$\left(\mathbf{J}_e \tilde{\mathbf{M}}_e^{-1} \mathbf{J}_e^T + \mathbf{U} \tilde{\mathbf{M}}_{\text{cond}}^{-1} \mathbf{V}^T \right) \lambda = -\tfrac{1+\varepsilon}{\Delta t} \mathbf{J} \mathbf{v}^-, \tag{12}$$

$$\mathbf{U} = \mathbf{J}_c - \mathbf{J}_e \tilde{\mathbf{M}}_e^{-1} \tilde{\mathbf{M}}_{ec}, \qquad \mathbf{V} = \mathbf{J}_c - \mathbf{J}_e \tilde{\mathbf{M}}_e^{-1} \tilde{\mathbf{M}}_{ce}^T.$$

We account for the rank-6 matrix $\mathbf{U} \tilde{\mathbf{M}}_{\text{cond}}^{-1} \mathbf{V}^T$ by applying a Sherman-Morrison-Woodbury update [GV96] to the solution of the full-rank linear system given by $\mathbf{J}_e \tilde{\mathbf{M}}_e^{-1} \mathbf{J}_e^T$. For the solution of the full rank system, we approximate $\tilde{\mathbf{M}}_e$ by considering only 3×3 block diagonal terms of the stiffness matrix $\mathbf{K}_e$. This approximation has the effect of discarding the implicit integration of inter-node elastic forces in the computation of the collision force. Note that the approximation still captures the two-way coupling of elastic forces between the core and the deformable layer, thereby preserving the responsiveness of the core's motion to collisions. Due to the block diagonal approximation of $\tilde{\mathbf{M}}_e$ and the structure of $\mathbf{J}_e$, where each constraint only affects one simulation node, the matrix $\mathbf{J}_e \tilde{\mathbf{M}}_e^{-1} \mathbf{J}_e^T$ is diagonal, and can be trivially inverted.

5.3.1. Moving Constraints

If a node i collides against a moving constraint $\mathbf{n}$, we estimate the world-frame velocity $\dot{\mathbf{x}}_o$ of the constraint at the time of maximum compression, and we rewrite the elastic collision equation (8) as:

$$\mathbf{n}^T \left(\dot{\mathbf{x}}_i^- + \frac{\delta \dot{\mathbf{x}}_i}{1+\varepsilon} - \dot{\mathbf{x}}_o \right) = 0, \tag{13}$$

To estimate $\dot{\mathbf{x}}_o$, we rigidify the colliding bodies and compute the normal velocity at the point of contact under a perfectly inelastic collision [Mir96].

5.3.2. Friction

We compute frictional response based on Coulomb's model, with friction coefficient μ_i for each colliding node i. Based on the kinematic relationship (19), pre-impact velocities $\mathbf{v}^-$, frictionless impulsive response $\delta\mathbf{v}$, a constraint normal $\mathbf{n}$, and pre-impact tangential velocity $\dot{\mathbf{x}}_i^t = \mathbf{L}_i \mathbf{v}^- - \left(\mathbf{n}^T \mathbf{L}_i \mathbf{v}^- \right) \mathbf{n}$,

we compute a maximally dissipating friction impulse $\delta\dot{\mathbf{x}}_i^t$ for node i similar to [BFA02] as:

$$\delta\dot{\mathbf{x}}_i^t = -\hat{\mu}\dot{\mathbf{x}}_i^t, \qquad \hat{\mu} = \min\left(1, \frac{\mu_i \|\mathbf{L}_i \delta\mathbf{v}\|}{\|\dot{\mathbf{x}}_i^t\|}\right). \tag{14}$$

We conclude by applying a friction impulse to the elastic velocity of the colliding node as $\delta\dot{\mathbf{q}}_i^t = \mathbf{R}^T \delta\dot{\mathbf{x}}_i^t$.

5.3.3. Constraint Correction

After the computation of collision response, we perform a position update with the newly computed velocities. With a new texture-based collision detection step, we detect possible colliding nodes and their penetration depth d. For a colliding pair of nodes i and j, with estimate local stiffness k_i and k_j, we determine the constraint position correction of node i to be $\delta\mathbf{x}_i = \frac{-k_j}{k_i+k_j} d\mathbf{n}$. Then, we correct the body-frame displacement of node i as $\delta\mathbf{q}_i = \mathbf{R}^T \delta\mathbf{x}_i$.

6. Algorithm and Parallel Implementation

In Figure 5 we outline the entire algorithm for simulating deformable objects in contact using dynamic deformation textures. Let s denote the operations that are performed on small-sized systems (i.e., computations of core variables, and low resolution collision detection). The remaining operations are all executed in a parallel manner on a large number of simulation nodes. Specifically, T refers to operations to be executed on all simulation nodes in the dynamic deformation texture T, D refers to operations to be executed on texels of the contact plane D, and T_D refers to operations to be executed on the colliding nodes.

As highlighted in Figure 5, all operations to be executed on simulation nodes can be implemented with parallelizable computation stencils. Even though our algorithm is valid for irregular meshings of the deformable layer, we chose a regular tetrahedral meshing for efficiency and for ease of implementation as it makes the computation stencils uniform across all nodes. Our parameterization scheme minimizes stretch [SSGH01] but requires special treatment at the patch boundaries in order to allow the use of uniform stencils. We adapt a method by Stam [Sta03] for providing accessible data in an 8-neighborhood to all nodes located on patch boundaries. Before every sparse matrix multiplication step in the algorithm, we fill a $\sqrt{2}$-texel-width region on patch boundaries by sampling values on the adjacent patches. Note that this sampling step removes the perfect symmetry of the large sparse linear systems to be solved, but we have not found problems in the convergence of the Conjugate Gradient solver.

Our simulation algorithm is especially well suited for parallel implementation. We designed a proof-of-concept implementation on a graphics processing unit (GPU). Dynamic deformation textures map naturally to GPU textures. We store and update the elastic displacements $\mathbf{q}_e$ and velocities $\mathbf{v}_e$ associated with FEM nodes in the deformable layer directly on the GPU. Similarly, we exploit the implicit FEM

	COLLISION-FREE UPDATE	
1.	Evaluate forces	T
2.	Solve the sparse linear systems $\tilde{\mathbf{M}}_e\mathbf{y} = \tilde{\mathbf{F}}_e$ and $\tilde{\mathbf{M}}_e\mathbf{Y} = \tilde{\mathbf{M}}_{ec}$ (Section 4.2), using a Conjugate Gradient solver [GV96]	T
3.	Update core velocities $\mathbf{v}_c^-$ using the condensed formulation (5)	s
4.	Update elastic velocities $\mathbf{v}_e^-$ using the new core velocities as in (6)	T
5.	Perform a position update $\mathbf{q}^- = \mathbf{q}(t) + \Delta t\mathbf{P}^+\mathbf{v}^-$	T
	COLLISION DETECTION	
6.	Execute low-resolution collision detection	s
7.	Execute high-resolution collision detection	D
8.	Map contact information to the dynamic deformation textures	T
	COLLISION RESPONSE	
9.	Invert the block-diagonalized full-rank matrix $\mathbf{J}_e\tilde{\mathbf{M}}_e^{-1}\mathbf{J}_e^T$ (Section 5.3)	T_D
10.	Solve for λ in (12) using the Sherman-Morrison-Woodbury formula	T_D
11.	Repeat steps 3 and 4 to obtain the collision impulse $\delta\mathbf{v}$, based on (10)	
12.	Compute friction impulse	T_D
13.	Perform a position update $\mathbf{q}(t+\Delta t) = \mathbf{q}^- + \Delta t\mathbf{P}^+(\delta\mathbf{v})$	T
	CONSTRAINT CORRECTION	
14.	Repeat collision detection steps 6 to 8	
15.	Apply constraint correction	T_D

Figure 5: ***Summary of the Simulation Algorithm***

mesh defined by the regular sampling of dynamic deformation textures to store the stiffness coefficients local to the nodes in auxiliary GPU textures. Elastic displacements and velocities (marked with T and T_D in the algorithm in Fig. 5) are updated in fragment programs on the GPU, using Frame Buffer Objects for direct computations on textures. The updates of core velocities, on the other hand, are executed on the CPU after gathering intermediate computations, which are performed in parallel on all nodes. Note that we minimize communication between CPU and GPU. The amount of data transfer is determined by the number of DoFs of the core. Our fast GPU implementation of the Conjugate Gradient solver exploits fast sparse matrix multiplications as demonstrated by [KW03] previously.

The collision detection stage also exploits image-based computations on the GPU. Per-texel penetration depth and contact normals are computed by orthonormal projection of the low-resolution core geometry onto the contact plane D, while texture mapping the positions of the high-resolution surfaces (Section 5.1). We use the same orthonormal projection and texture matrix to map the contact data from the contact plane D back to the dynamic deformation texture T for contact response, a projection technique similar to shadow mapping.

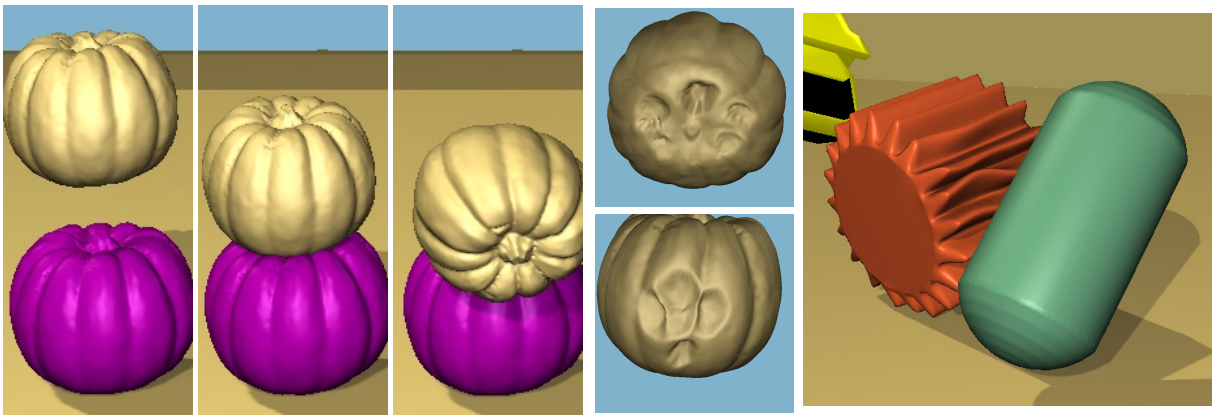

Figure 6: ***Deformations of High-Resolution Geometry.*** *Left: Two deformable pumpkins are dropped on top of each other. Center: Detail view of the rich deformations produced on the top pumpkin during contact. Right: A dropped cylinder produces rich dynamic deformations on the ridges of a gear.*

7. Results and Discussion

We have executed our experiments on a 3.4 GHz Pentium-4 processor PC with a Nvidia GeForce 7800GTX graphics card. Table 1 lists the statistics of the main models we have used. In all cases, we have employed dynamic deformation textures of 256×256 texels. Such high resolution enables the simulation of rich deformations, as shown most evidently in the pumpkins (Figure 6), gear (Figure 6), and head (Figure 7) models. With our constraint-based collision response approach, impacts produce highly-detailed indentations such as the ones suffered when the pumpkins are dropped on each other, or when the fist punches the head on the eyebrow. The gear model demonstrates rich dynamic deformations of surface features larger than 30% of the object radius.

Model	Tire	Cylinder	Pumpkin	Gear	Head
Nodes	31K	21K	30K	29K	40K
Tets.	162K	161K	183K	173K	240K

Table 1: ***Models and Statistics.***

The use of sound physically-based techniques for modeling contact and deformations leads to highly plausible rolling and tumbling motion in combination with surface deformations, as can be perceived in the tires (Figure 1) and gear (Figure 6) scenes. With our deformable object representation and simulation algorithm, we achieve those effects on high-resolution objects in an efficient manner. The computational cost is dominated by the iterative solver, and its convergence depends mostly on the stiffness of the regions in contact. As a reference, the simulation of the rolling heterogeneous cylinder depicted in Figure 3 runs at an average of $1-2$ fps when the stiff part (Young modulus 60KN/m^2) is in contact with the ridged terrain. On the other hand, the same simulation runs at an average of 6 fps when the soft part (Young modulus 3KN/m^2) is in contact, with up to 2600 simultaneously colliding nodes. This translates to a throughput of approximately 1M tetrahedra and 120K surface simulation nodes simulated per second. In the rest of the experiments, we reach similar average performance: 2 seconds/frame for the simulation of the tires (Figure 1), 1 fps for the punch (Figure 7) and cylinder-with-gear (Figure 6) scenes, and 2 fps for the dropped head (Figure 7).

Method	DoFs	Contact	Performance
BNC96	surface	explicit	11K nodes/sec
ZC99	volume	explicit	303K els./sec
PPG04	surface	LCP	2K contacts/sec
MG04	volume	explicit	63K els./sec
D^{2}T	surface	implicit Lagrange mult.	120K nodes./sec 1M els./sec 15K contacts/sec

Table 2: ***Approximate Performance Data Benchmark.*** *Extrapolated performance data from [BNC96], [ZC99], [PPG04], [MG04] shown with ours,* ***D^{2}T****.*

We have chosen a few related techniques as a basis for benchmarking the overall performance of our algorithm. It is, however, very difficult to compare the various techniques, as their primary goals are often different. Our algorithm complements the prior work by offering an efficient, robust contact handling method for colliding deformable bodies with *large contact regions and high-resolution surface geometry*, but cannot simulate arbitrary large deformations. We have extrapolated performance data using Moore's Law (performance increases 2x every 18 months). As indicated in Table 2, the performance of our approach (D^{2}T) (up to 15K contacts/sec, 1M tets/sec, and 120K nodes/sec for moderately soft objects), is comparable to the performance of techniques that use explicit integration (e.g. [ZC99]), without their time-step restrictions. Our approach is considerably faster than other methods that enable large time steps, both those that focus on the surface deformation (such as [BNC96]), and efficient corotational methods that compute deformations within the entire volume (such as [MG04]). Our approach can also handle many more contact points than novel quasi-rigid dynamics algorithms using LCP [PPG04], producing richer deformations. Though

computationally very efficient, it cannot achieve a performance comparable to model reduction techniques that precompute data-driven models and build efficient low-rank approximations of deformed shapes [BJ05]. On the other hand, our approach does not require lengthy pre-computation of dynamics and achieves rich high-resolution deformations with both local and global support.

8. Summary and Future Work

We have presented a fast and robust physically-based approach for dynamic simulation of colliding deformable bodies with high-resolution surface geometry and large contact regions. We have overcome the computational challenges of this problem by designing a novel layered representation of deformable models, where surface layer deformations are represented and computed in a reduced, two-dimensional parametric domain. We have also presented efficient solutions for collision detection, physically-based simulation of dynamic deformations, and robust contact response, by exploiting the layered representation of the models and decoupling the degrees of freedom between the core and the deformation layers. To conclude, we have demonstrated the implementation of our algorithm on parallel processors, achieving fast simulation of demanding scenarios with detailed deformations and thousands of simultaneous collisions.

The use of a layered representation obviously poses some limitations on the type of deformations that can be modeled. Nevertheless, we have successfully captured large deformations that reach as much as 30-40% of the radii of objects. Moreover, our representation can be extended to articulated, flexible bodies that undergo skeletal deformations, by augmenting the generalized coordinate set of the core representation to include multibody systems. Extending the formulation to richer core configurations or multiple deformable layers may also require novel methods for collision detection, as distortion problems may become more pronounced. The two step collision detection process could be enhanced by incorporating low-resolution deformable objects and handling of self-collision.

With the current framework and future enhancements, we plan to apply our algorithm and representation to the simulation of characters with skeletal control and plausible collision avoidance. Further optimizations of the algorithm and the steady growth of parallel processors offer possibilities for interactive, detailed physically-based simulation of skin deformations in character animation.

Acknowledgements. This work is supported in part by ARO, NSF, ONR, DARPA/RDECOM and the Swiss NSF. We'd like to thank the GAMMA group in Chapel Hill and the CGL in Zurich for their support; in particular Avneesh Sud, Denis Steinemann and Martin Wicke for early reviews and critical input, Jason Sewall for last-minute video editing and Mark Harris (NVIDIA) for GPGPU support.

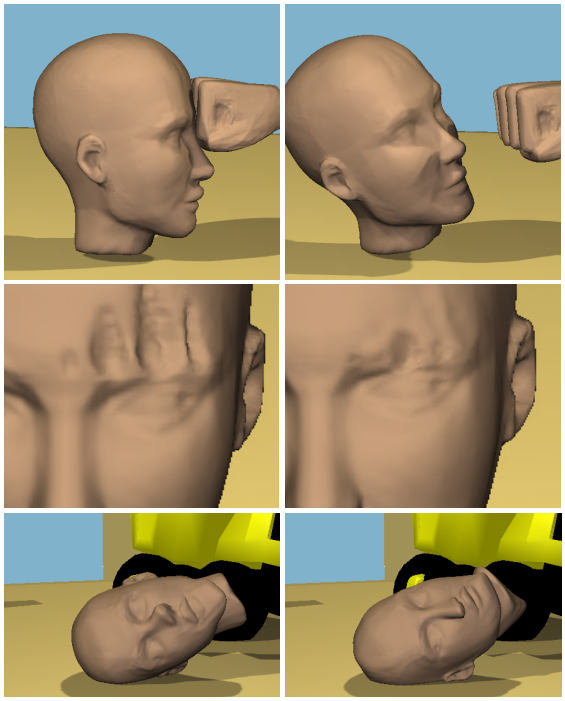

Figure 7: ***Deformations of a Virtual Head.*** *Top: A fist hits a deformable head (attached by springs in the neck area), producing both local deformations and global motion. Middle: Detail of the deformations produced near the eyebrow by the impact. Bottom: A softer head is dropped on the floor, resulting in larger deformations.*

References

[BFA02] BRIDSON R., FEDKIW R., ANDERSON J.: Robust treatment of collisions, contact and friction for cloth animation. In *Proc. of ACM SIGGRAPH* (2002).

[BJ05] BARBIČ J., JAMES D. L.: Real-time subspace integration of St. Venant-Kirchhoff deformable models. In *Proc. of ACM SIGGRAPH* (2005).

[BNC96] BRO-NIELSEN M., COTIN S.: Real-time volumetric deformable models for surgery simulation using finite elements and condensation. *Computer Graphics Forum 15*, 3 (1996).

[CBC*05] CAPELL S., BURKHART M., CURLESS B., DUCHAMP T., POPOVIC Z.: Physically based rigging for deformable characters. *Proc. of Eurographics/ACM SIGGRAPH Symposium on Computer Animation* (2005).

[CGC*02] CAPELL S., GREEN S., CURLESS B., DUCHAMP T., POPOVIC Z.: A multiresolution framework for dynamic deformations. *Proc. of ACM SIGGRAPH Symposium on Computer Animation* (2002).

[CHP89] CHADWICK J. E., HAUMANN D. R., PARENT R. E.: Layered construction for deformable animated characters. In *Proc. of ACM SIGGRAPH* (1989).

[COM98] COHEN J., OLANO M., MANOCHA D.: Appearance-preserving simplification. In *Proc. of ACM SIGGRAPH* (1998).

[CW05] CIRAK F., WEST M.: Decomposition contact response (DCR) for explicit finite element dynamics. *International Journal for Numerical Methods in Engineering 64*, 8 (2005).

[DAK04] DURIEZ C., ANDRIOT C., KHEDDAR A.: Signorini's contact model for deformable objects in haptic simulations. *Proc. of IEEE/RSJ IROS* (2004).

[DDCB01] DEBUNNE G., DESBRUN M., CANI M. P., BARR A. H.: Dynamic real-time deformations using space and time adaptive sampling. *Proc. of ACM SIGGRAPH* (2001).

[EL01] EHMANN S. A., LIN M. C.: Accurate and fast proximity queries between polyhedra using convex surface decomposition. In *Proc. of Eurographics* (2001).

[Gas98] GASCUEL M.-P.: Layered deformable models with implicit surfaces. In *Proc. of Graphics Interface* (1998).

[GKS02] GRINSPUN E., KRYSL P., SCHRÖDER P.: CHARMS: A simple framework for adaptive simulation. *Proc. of ACM SIGGRAPH* (2002).

[GPS02] GOLDSTEIN H., POOLE C., SAFKO J.: *Classical Mechanics, 3rd Ed.* Addison Wesley, 2002.

[GV96] GOLUB G. H., VAN LOAN C. F.: *Matrix Computations*, 3rd ed. Johns Hopkins University Press, 1996.

[HFS03] HIROTA G., FISHER S., STATE A.: An improved finite element contact model for anatomical simulations. *The Visual Computer 19*, 5 (2003).

[HSO03] HAUSER K. K., SHEN C., O'BRIEN J. F.: Interactive deformation using modal analysis with constraints. *Proc. of Graphics Interface* (2003).

[JP99] JAMES D. L., PAI D. K.: ArtDefo: accurate real time deformable objects. In *Proc. of ACM SIGGRAPH* (1999).

[JP02] JAMES D. L., PAI D. K.: DyRT: Dynamic response textures for real-time deformation simulation with graphics hardware. In *Proc. of ACM SIGGRAPH* (2002).

[KW03] KRUGER J., WESTERMANN R.: Linear algebra operators for GPU implementation of numerical algorithms. In *Proc. of ACM SIGGRAPH* (2003).

[MG04] MÜLLER M., GROSS M.: Interactive virtual materials. *Proc. of Graphics Interface* (2004).

[Mir96] MIRTICH B. V.: *Impulse-Based Dynamic Simulation of Rigid Body Systems*. PhD thesis, University of California at Berkeley, 1996.

[MT92] METAXAS D., TERZOPOULOS D.: Dynamic deformation of solid primitives with constraints. *Proc. of ACM SIGGRAPH* (1992).

[NMK*05] NEALEN A., MÜLLER M., KEISER R., BOXERMANN E., CARLSON M.: Physically based deformable models in computer graphics (state of the art report). *Eurographics STAR* (2005).

[OJSL04] OTADUY M. A., JAIN N., SUD A., LIN M. C.: Haptic display of interaction between textured models. In *Proc. of IEEE Visualization* (2004).

[PPG04] PAULY M., PAI D. K., GUIBAS L. J.: Quasi-rigid objects in contact. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2004).

[Sha89] SHABANA A. A.: *Dynamics of Multibody Systems*. John Wiley and Sons, 1989.

[SOH99] SUMNER R. W., O'BRIEN J. F., HODGINS J. K.: Animating sand, mud, and snow. *Computer Graphics Forum 18*, 1 (1999).

[SSGH01] SANDER P. V., SNYDER J., GORTLER S. J., HOPPE H.: Texture mapping progressive meshes. *Proc. of ACM SIGGRAPH* (2001).

[Sta03] STAM J.: Flow on surfaces of arbitrary topology. In *Proc. of ACM SIGGRAPH* (2003).

[TKH*05] TESCHNER M., KIMMERLE S., HEIDELBERGER B., ZACHMANN G., RAGHUPATHI L., FURHMANN A., CANI M.-P., FAURE F., MAGNENAT-THALMANN N., STRASSER W., VOLINO P.: Collision detection for deformable objects. *Computer Graphics Forum 24*, 1 (2005).

[TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically deformable models. *Proc. of ACM SIGGRAPH* (1987).

[TSIF05] TERAN J., SIFAKIS E., IRVING G., FEDKIW R.: Robust quasistatic finite elements and flesh simulation. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2005).

[TW88] TERZOPOULOS D., WITKIN A.: Physically based models with rigid and deformable components. *IEEE Computer Graphics and Applications 8*, 6 (1988).

[WRM05] WROTEK P., RICE A., MCGUIRE M.: Real-time collision deformations using graphics hardware. *Journal of Graphics Tools 10*, 5 (2005).

[WVS90] WRIGGERS P., VU VAN T., STEIN E.: Finite element formulation of large deformation impact-contact problems with friction. *Computers & Structures 37*, 3 (1990).

[ZC99] ZHUANG Y., CANNY J.: Real-time simulation of physically realistic global deformation. *Proc. of IEEE Visualization* (1999).

[ZSS97] ZORIN D., SCHRÖDER P., SWELDENS W.: Interactive multiresolution mesh editing. In *Proc. of ACM SIGGRAPH* (1997).

Appendix A: Kinematic Relationships

The angular velocity $\omega \in \mathbb{R}^3$ can be expressed in terms of the derivative of a quaternion $\theta \in \mathbb{R}^4$ by the linear relationship $\omega = \mathbf{G}\dot{\theta}$ [Sha89]. Similarly, we can express the relationship between the velocity state vector $\mathbf{v}$ and the time-derivatives of the generalized coordinates $\mathbf{q}$ as:

$$\mathbf{v} = \mathbf{P}\dot{\mathbf{q}}, \qquad \dot{\mathbf{q}} = \mathbf{P}^{+}\mathbf{v}, \tag{15}$$

$$\mathbf{P} = \begin{pmatrix} \mathbf{I}_3 & 0 & 0 \\ 0 & \mathbf{G} & 0 \\ 0 & 0 & \mathbf{I}_n \end{pmatrix}, \qquad \mathbf{P}^{+} = \begin{pmatrix} \mathbf{I}_3 & 0 & 0 \\ 0 & \frac{1}{4}\mathbf{G}^T & 0 \\ 0 & 0 & \mathbf{I}_n \end{pmatrix},$$

where n is the number of elastic coordinates, and $\mathbf{P}\mathbf{P}^{+} = I$.

We can now derive the world-frame velocity of a material point in terms of the velocity state vector:

$$\dot{\mathbf{x}} = \dot{\mathbf{c}} + \dot{\mathbf{R}}\mathbf{u} + \mathbf{R}\dot{\mathbf{u}} \tag{16}$$

$$= \dot{\mathbf{c}} + \mathbf{B}\dot{\theta} + \mathbf{R}\mathbf{S}\dot{\mathbf{q}}_e. \tag{17}$$

The matrix $\mathbf{B}$ is the Jacobian of the vector $\mathbf{R}\mathbf{u}$ w.r.t. θ, and it can be proven to be equal to $-\mathbf{R}\tilde{\mathbf{u}}\mathbf{G}$ [Sha89], where $\tilde{\mathbf{u}}$ is the skew-symmetric cross-product matrix obtained from $\mathbf{u}$. We can rewrite (17) in compact matrix form as a linear function of the time derivative of the generalized coordinate set $\mathbf{q}$:

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{I}_3 & -\mathbf{R}\tilde{\mathbf{u}}\mathbf{G} & \mathbf{R}\mathbf{S} \end{bmatrix} \dot{\mathbf{q}}. \tag{18}$$

Applying $\dot{\mathbf{q}} = \mathbf{P}^{+}\mathbf{v}$, we can rewrite (18) and obtain:

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{I}_3 & -\mathbf{R}\tilde{\mathbf{u}} & \mathbf{R}\mathbf{S} \end{bmatrix} \mathbf{v} = \mathbf{L}\mathbf{v}. \tag{19}$$