Multiresolution Green's Function Methods for Interactive Simulation of Large-Scale Elastostatic Objects

DOUG L. JAMES and DINESH K. PAI
University of British Columbia

We present a framework for low-latency interactive simulation of linear elastostatic models, and other systems arising from linear elliptic partial differential equations, which makes it feasible to interactively simulate large-scale physical models. The deformation of the models is described using precomputed Green's functions (GFs), and runtime boundary value problems (BVPs) are solved using existing Capacitance Matrix Algorithms (CMAs). Multiresolution techniques are introduced to control the amount of information input and output from the solver thus making it practical to simulate and store very large models. A key component is the efficient compressed representation of the precomputed GFs using second-generation wavelets on surfaces. This aids in reducing the large memory requirement of storing the dense GF matrix, and the fast inverse wavelet transform allows for fast summation methods to be used at run-time for response synthesis. Resulting GF compression factors are directly related to interactive simulation speedup, and examples are provided with hundredfold improvements at modest error levels. We also introduce a multiresolution constraint satisfaction technique formulated as a hierarchical CMA, so named because of its use of hierarchical GFs describing the response due to hierarchical basis constraints. This direct solution approach is suitable for hard real-time simulation since it provides a mechanism for gracefully degrading to coarser resolution constraint approximations. The GFs' multiresolution displacement fields also allow for run-time adaptive multiresolution rendering.

Categories and Subject Descriptors: E.4 [**Coding and Information Theory**]: Data Compaction and Compression; F.2.1 [**Numerical Algorithms and Problems**]: Computation of Transforms; G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*linear systems, matrix inversion*; G.1.9 [**Numerical Analysis**]: Integral Equations; I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*physically based modeling*; I.3.6 [**Computer Graphics**]: Methodology and Techniques—*interaction techniques*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*animation, Virtual reality*

General Terms: Algorithms

Additional Key Words and Phrases: Capacitance matrix, deformation, elastostatic, fast summation, force feedback, Green's function, interactive real-time applications, lifting scheme, wavelets, real-time, updating

1. INTRODUCTION

Interactive multimodal simulation of deformable objects, in which a user may manipulate flexible objects and receive immediate sensory feedback via human–computer interfaces, is a major challenge for computer graphics and virtual environments. Deformation is essential in computer animation for plausibly modeling the behavior of the human body, animals, and soft objects such as furniture upholstery, but interactive applications, such as computer games, have very limited computing budgets for 3D physical continuum simulation. Current virtual prototyping and assembly planning applications require interactive simulations of deformable and also flexible kinematic models with complex geometry. Deformable models have a long history (see Section 2) and, one might say, are well understood within the graphics, scientific, and engineering communities. The challenge addressed here is the design of

Authors' address: Department of Computer Science, University of British Columbia, 2366 Main Mall, Vancouver, B.C. V6T 1Z4, Canada; email: {djames,pai}@cs.ubc.ca.

deformable models that are both sufficiently realistic to capture the relevant physics, and sufficiently fast for interactive simulation.

In recent years, *linear elastostatic Green's function models* (LEGFMs) have been shown to strike an attractive trade-off between realism and speed. The models are physically based and are accurate approximations of small-strain elastic deformations for objects in equilibrium. In practice, they are an appealing model for simulating deformable materials that are relatively stiff (with respect to applied forces) and tend to reach equilibrium quickly during continuous contact. The linearity of the model allows for the use of extremely fast solution algorithms based on linear superposition that support real-time rendering and stable force feedback. The use of these techniques in interactive simulation was advanced by Bro-Nielsen and Cotin [1996], Cotin et al. [1999], and James and Pai [1999], who demonstrated real-time interaction with deformable models in applications such as force feedback surgical simulation and computer animation. Reality-based active measurement techniques also exist for acquisition of LEGFMs [Pai et al. 2001].

The key to the fast simulation technique is a data-driven formulation based on precomputed *Green's functions* (GFs). GFs provide a natural data structure for subsuming details of model creation such as numerical discretization or measurement and estimation. Intuitively, GFs form a basis for representing all possible deformations of an object in a particular geometric configuration of boundary constraint types, for example, essential (Dirichlet), natural (Neumann), or mixed (Robin). The benefit of linearity is that the response to any set of boundary values can be quickly reconstructed by a linear combination of precomputed GFs. In this way, these solution techniques can be used to obtain the solution for any set of applied constraints by using the GFs in combination with a collection of matrix updating methods (related to the Sherman–Morrison–Woodbury formula) which we refer to collectively as Capacitance Matrix Algorithms (CMAs).

Since general linear systems principles are exploited, the GF-based CMA matrix solvers are not just limited to LEGFMs, and can in fact be used to simulate numerous other continuous physical systems in equilibrium, such as those described by linear elliptic partial differential equations (PDEs), for example, modeling electrostatic fields, equilibrium diffusion, and transport phenomena. An interesting point is that LEGFMs are small strain approximations of finite strain elasticity, however, other physical systems are accurately modeled by linear elliptic PDEs such as electrostatics. So although this article is presented in the context of deformable object simulation, parallel relationships exist between physical quantities in other applications.

The CMAs achieve their fast visualization speed at the expense of storing $O(n^2)$ elements of the large dense GF matrices that can be accessed in constant time. This clearly does not scale well to large models; for example, the GF matrix stored as floats for a vertex-based model with 100 vertices requires only 360 KB, however, one with 10,000 vertices, such as the dragon model in Figure 8, would require 3.6 GB! For these problems bus and cache considerations are significant as well. In addition, the CMA presented in James and Pai [1999] requires an $O(s^3)$ factoring of the dense capacitance matrix, which scales very poorly as the number of run-time constraints increases.

In this article we present a family of algorithms for simulating deformable models and related systems that make GF techniques practical for very large models. The multiresolution techniques do much more than simply compress GFs to minimize storage. As a rule, these approaches are compatible with and improve the performance and real-time feasibility of numerical operations required for the direct solution of boundary value problems. The algorithms exploit the fact that there exist several distinct, yet related, spatial scales corresponding to

—geometric detail,
—elastic displacement fields,

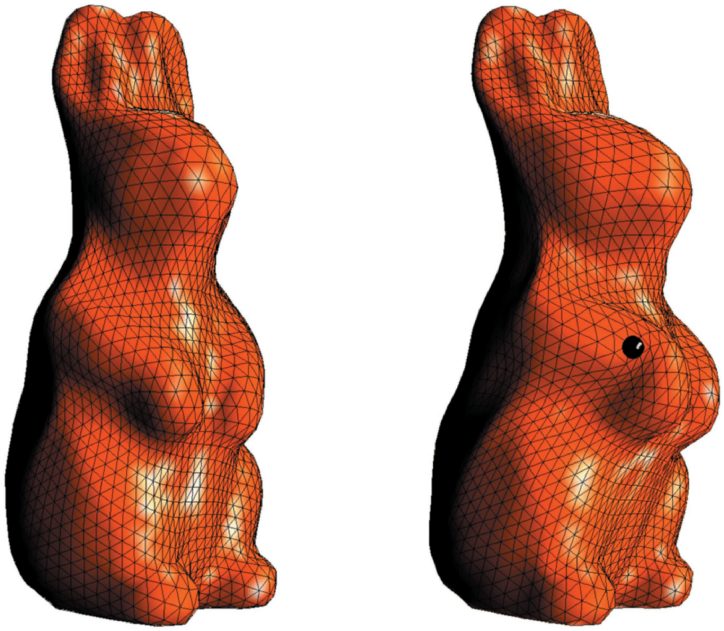

Fig. 1. Preview of results for a complex elastic model. An elastic rabbit model with 2562 vertices, 5120 faces, and 5 levels of subdivision connectivity ($L = 4$), capable of being rendered at 30 FPS with 1 kHz force feedback on a PC in our Java-based haptic simulation. The associated dense square Green's function submatrix contained 41 million floats (166 MB) but was compressed down to 655 thousand floats (2.6 MB) in this animation ($\varepsilon = .2$). The depicted deformation resulted from force interactions defined at a constraint resolution that was two levels coarser (L = 2) than the visible mesh; for these coarse level constraints, the GF matrix block may be further compressed by a factor of approximately $16 = 4^2$. Even further compression is possible with file formats for storage and transmission of models. (Reparameterized rabbit model generated from mesh courtesy of Cyberware)

—elastic traction fields, and

—numerical discretization.

We develop multiresolution summation techniques to quickly synthesize deformations, and hierarchical CMAs to deal with constraint changes. Wavelet GF representations are also useful for simulating multiresolution geometry for graphical and haptic rendering. For a preview of our results see Figure 1.

2. RELATED WORK

Substantial work has appeared on physical deformable object simulation and animation in computer graphics and related scientific fields [Terzopoulos et al. 1987; Baraff and Witkin 1992; Gibson and Mirtich 1997; Cotin et al. 1999; Zhuang and Canny 2000; Debunne et al. 2001], although not all is suited for interactive applications. Important applications for interactive elastic simulation include computer animation and interactive games, surgical simulation, computer-aided design, interactive path planning, and virtual assembly and maintenance for increasingly complicated manufacturing processes.

There has been a natural interactive simulation trend toward explicit temporal integration of (lumped mass) nonlinear FEM systems, especially for biomaterials undergoing large strains, with examples using parallel computation [Székely et al. 2000], adaptivity in space [Zhuang and Canny 2000], and

perhaps time [Debunne et al. 2001; Wu et al. 2001], and also adaptive use of linear and nonlinear elements [Picinbono et al. 2001]. However, several limitations can be overcome for LEGFMs: (1) only several hundred interior nodes can be integrated at a given time (without special hardware), and (2) although these are excellent models for soft materials, numerical stiffness can make it costly to timestep explicit models that are, for example, physically stiff, incompressible, or have detailed discretizations.

Implicit integration methods [Ascher and Petzold 1998] (appropriate for numerically stiff equations) have been revived in graphics [Terzopoulos and Fleischer 1988; Hauth and Etzmuss 2001] and successfully applied to offline cloth animation [Baraff and Witkin 1998]. These integrators are generally not used for large-scale *interactive* 3D models due to the cost of solving a large linear system each timestep (however, see Desbrun et al. [1999] and Kang et al. [2000] for cloth models of moderate complexity). Multirate integration approaches are useful for supporting haptic interactions with timestepped models [Astley and Hayward 1998; Çavuşoğlu and Tendick 2000; Balaniuk 2000; Debunne et al. 2001].

Modal analysis for linear elastodynamics [Pentland and Williams 1989] is effective for simulating free vibration (as opposed to continuous contact interactions), and has been used for interactive [Stam 1997], force feedback [Basdogan 2001], and contact sound simulation [van den Doel and Pai 1998] by precomputing or measuring modal data (see also *DyRT* [James and Pai 2002a]). Related dimensional reduction methods exist for nonlinear dynamics [Krysl et al. 2001].

Boundary integral formulations for linear elastostatics have well-understood foundations in potential theory [Kellogg 1929; Jaswon and Symm 1977] and are based on, for example, singular free-space Green's function solutions of Navier's equation for which analytic expressions are known. On the other hand, precomputed linear elastostatic models for real-time simulation use numerically derived discrete Green's function solutions corresponding to particular geometries and constraint configurations, and are also not restricted to homogeneous and isotropic materials. These approaches are relatively new [Bro-Nielsen and Cotin 1996; Hirota and Kaneko 1998; Cotin et al. 1999; James and Pai 1999; Berkley et al. 1999; James and Pai 2001, 2002b], and yet used in, for example, commercial surgical simulators [Kühnapfel et al. 1999]. Prior to real-time applications, related ideas in matrix updating for elliptic problems were not uncommon [Proskurowski and Widlund 1980; Kassim and Topping 1987; Hager 1989]. Our previous work on real-time Green's function simulation, including the ArtDefo simulator for interactive computer animation [James and Pai 1999] and real-time haptics [James and Pai 2001], was initially inspired by pioneering work in boundary element contact mechanics [Ezawa and Okamoto 1989; Man et al. 1993]. We derived capacitance matrix updating equations in terms of GFs (directly from the BEM matrix equations in James and Pai [1999]) using the Sherman–Morrison–Woodbury formulae, and provided examples from interactive computer animation and haptics for distributed contact constraints. Of notable mention is the work on real-time laparoscopic hepatic surgery simulation by the group at INRIA [Bro-Nielsen and Cotin 1996; Cotin et al. 1999], in which pointlike boundary displacement constraints are resolved by determining the correct superposition of precomputed GF-like quantities is identifiable as a special case of the CMA. This article addresses the fact that *all of these approaches suffer from poorly scaling precomputation and memory requirements which ultimately limit the complexity of models that can be constructed and/or simulated.*

We make extensive use of multiresolution modeling related to subdivision surfaces [Loop 1987; Zorin and Schröder 2000] and their displaced variants [Lee et al. 2000]. Our multiresolution elastostatic surface splines also have connections with variational and physically based subdivision schemes [Dyn et al. 1990; Weimer and Warren 1998, 1999]. We are mostly concerned with the efficient manipulation of GFs defined on (subdivision) surfaces. Natural tools here are subdivision wavelets [Lounsbery et al. 1997], and we make extensive use of biorthogonal wavelets based on the lifting scheme [Sweldens 1998; Schröder and Sweldens 1995a,b] for efficient GF represention, fast summation, and hierarchical

constraint bases generation [Yserentant 1986]. Efficient represention of functions on surfaces [Kolarov and Lynch 1997] is also related to the larger area of multiresolution and progressive geometric representation; for example, see Khodakovsky et al. [2000].

Our work on wavelet GFs is related to multiresolution discretization techniques [Beylkin et al. 1991b; Alpert et al. 1993] for sparse represention of integral operators and fast matrix multiplication. Unlike cases from classical potential theory where the integral operator's kernel is analytically known, for example, free-space GF solutions [Jaswon and Symm 1977], and can be exploited [Greengard and Rokhlin 1987; Hackbusch and Nowak 1989; Yoshida et al. 2001], or for wavelet radiosity in which the form factors may be extracted relatively easily [Gortler et al. 1993], here the integral operator's discrete matrix elements are defined implicitly as discrete GFs obtained by numerical solution of a class of boundary value problems (BVPs). Nevertheless, it is known that such (GF) integral operators have sparse represention in wavelet bases [Beylkin 1992]. Representation restrictions are also imposed by CMA efficiency concerns.

Finally, the obvious approach to simulating large elastostatic models interactively is to just use standard numerical methods [Zienkiewicz 1977; Brebbia et al. 1984], and especially "fast" iterative solvers such as multigrid [Hackbusch 1985] for domain discretizations, and preconditioned fast multipole [Greengard and Rokhlin 1987; Yoshida et al. 2001] or fast wavelet transform [Beylkin 1992] methods for boundary integral discretizations. Such methods are highly suitable for GF precomputation, but we do not consider them suitable for online interactive simulation; our experience with large models has been that these methods can be several orders of magnitude slower than the methods presented herein (e.g., see §7.6.2 of James [2001] for speedups of over 100,000 times). Worse still, online methods fail to provide fast (random) access to GF matrix elements, for example, for haptics, output-sensitive selective simulation, and the loss of the GF data abstraction destroys our ability to immediately simulate scanned physical data sets [Pai et al. 2001].

3. BACKGROUND: INTERACTIVE SIMULATION OF GREEN'S FUNCTION MODELS USING MATRIX UPDATING TECHNIQUES

3.1 Linear Elastostatic Green's Function Models

Linear elastostatic objects are generalized three-dimensional linear springs, and as such they are useful modeling primitives for physically based simulations. In this section, background material for a generic discrete Green's function description for precomputed linear elastostatic models is provided. It is not an introduction to the topic, and the reader might consult a suitable background reference before continuing [Barber 1992; Hartmann 1985; Zienkiewicz 1977; Brebbia et al. 1984]. The GFs form a basis describing all possible deformations of a linear elastostatic model subject to a certain class of constraints. This is useful because it provides a common language to describe all discrete models and subsumes extraneous details regarding discretization or measurement origins.

Another benefit of using GFs is that they provide an efficient means for exclusively simulating only boundary data (displacements and tractions). This is useful when rendering of interior data is not required or in cases where it may not even be available, such as for reality-based models obtained via boundary measurement [Pai et al. 2001]. Although it is possible to simulate various internal volumetric quantities (Section 3.1.3), simulating only boundary data involves less computation. This is sufficient since in interactive computer graphics we are primarily concerned with interactions that impose surface constraints and provide feedback via visible surface deformation and contact forces.

3.1.1 *Geometry and Material Properties.* Given that the fast solution method is based on linear systems principles, essentially any linear elastostatic model with physical geometric and material properties is admissible. We consider models in three dimensions, although many arguments also apply

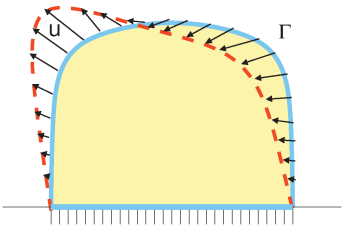

Fig. 2. Discrete nodal displacements u defined at nodes, for example, vertices, on the undeformed boundary Γ (solid blue line), that result in a deformation of the surface (to dashed red line). Although harder to illustrate, a similar definition exists for the traction vector p.

to lower dimensions. Suitable models would of course include bounded volumetric objects with various internal material properties, as well as special subclasses such as thin plates and shells. Since only a boundary or interface description is utilized for specifying user interactions, other exotic geometries may also be easily considered such as semi-infinite domains, exterior elastic domains, or simply any set of parameterized surface patches with a linear response. Similarly, numerous representations of the surface and associated displacement shape functions are also possible, such as, polyhedra, NURBS, and subdivision surfaces [Zorin and Schröder 2000].

Let the undeformed boundary be denoted by Γ. The change in shape of this surface is described by the surface *displacement field* $\mathbf{u}(\mathbf{x})$, $\mathbf{x} \in \Gamma$, and the surface force distribution is called the *traction field* $\mathbf{p}(\mathbf{x})$, $\mathbf{x} \in \Gamma$. Each is parameterized by n nodal variables (see Figure 2), so that the discrete displacement and traction vectors are

$$\mathsf{u} = [\mathsf{u}_1, \ldots, \mathsf{u}_n]^\mathsf{T} \tag{1}$$

$$\mathsf{p} = [\mathsf{p}_1, \ldots, \mathsf{p}_n]^\mathsf{T}, \tag{2}$$

respectively, where each nodal value is a 3-vector. The continuous traction field may then be defined as a 3-vector function

$$\mathbf{p}(\mathbf{x}) = \sum_{j=1}^{n} \phi_j(\mathbf{x}) \mathsf{p}_j, \tag{3}$$

where $\phi_j(\mathbf{x})$ is a scalar basis function associated with the jth node. The force on any surface area is equal to the integral of $\mathbf{p}(\mathbf{x})$ on that area. We can then define the nodal force associated with any nodal traction as

$$\mathsf{f}_j = a_j \mathsf{p}_j \quad \text{where} \quad a_j = \int_\Gamma \phi_j(\mathbf{x}) d\Gamma_\mathbf{x} \tag{4}$$

defines the area associated with the jth node. A similar space exists for the continuous displacement field components, and is in general different from the traction field.

Our implementation uses linear boundary element models, for which the nodes are vertices of a closed triangle mesh model using Loop subdivision [Loop 1987]. Such surfaces are convenient for obtaining multiresolution models for rendering as well as smoothly parameterized surfaces suitable for BEM discretization and deformation depiction. We describe both the traction field and the polyhedral displacement field using continuous piecewise linear basis functions: $\phi_j(\mathbf{x})$ represents a "hat function" located at the jth vertex normalized so that $\phi_j(\mathbf{x}_i) = \delta_{ij}$. Given our implementation, we often refer to node and vertex interchangeably. The displacement and traction fields both have convenient vertex-based descriptions

$$\mathsf{u}_j = \mathbf{u}(\mathbf{x}_j), \quad \mathsf{p}_j = \mathbf{p}(\mathbf{x}_j), \quad j = 1, \ldots, n, \tag{5}$$

where $\mathbf{x}_j$ is the jth vertex location.

3.1.2 *Discrete Boundary Value Problem.* At each step of the simulation, a discrete BVP must be solved that relates specified and unspecified nodal values, for example, to determine deformation and force feedback forces. Without loss of generality, it is assumed that either position or traction constraints are specified at each boundary node, although this can be extended to allow mixed conditions such as normal displacement and tangential tractions. Let nodes with prescribed displacement or traction constraints be specified by the mutually exclusive index sets Λ_u and Λ_p, respectively, so that $\Lambda_u \cap \Lambda_p = \emptyset$ and $\Lambda_u \cup \Lambda_p = \{1, 2, \ldots, n\}$. We refer to the (Λ_u, Λ_p) pair as the system constraint or *BVP type*. We denote the unspecified and complementary specified nodal variables by

$$\mathsf{v}_j = \begin{cases} \mathsf{p}_j : j \in \Lambda_u \\ \mathsf{u}_j : j \in \Lambda_p \end{cases} \quad \text{and} \quad \overline{\mathsf{v}}_j = \begin{cases} \mathsf{u}_j : j \in \Lambda_u \\ \mathsf{p}_j : j \in \Lambda_p \end{cases}, \tag{6}$$

respectively. Typical boundary conditions for a force feedback loop, for example, consist of specifying some (compactly supported) displacement constraints in the area of contact, with free boundary conditions (zero traction) and other (often zero displacement) support constraints outside the contact zone. In order to guarantee an equilibrium constraint configuration (hence elasto*static*) we must formally require at least one displacement constraint, $\Lambda_u \neq \emptyset$, to avoid an ambiguous rigid body translation.

3.1.3 *Fast BVP Solution with Green's Functions.* GFs for a *single BVP type* provide an economical means for solving *that particular BVP*, but when combined with the CMA (Section 3.2) the GFs can also be used to solve *other BVP types*. The general solution of a particular BVP type (Λ_u, Λ_p) may be expressed in terms of its discrete GFs as

$$\boxed{\mathsf{v} = \Xi\overline{\mathsf{v}} = \sum_{j=1}^{n} \xi_j \overline{\mathsf{v}}_j = \sum_{j \in \Lambda_u} \xi_j \mathsf{u}_j + \sum_{j \in \Lambda_p} \xi_j \mathsf{p}_j}, \tag{7}$$

where the discrete GFs of the particular BVP system are the block column vectors ξ_j assembled in the GF matrix

$$\boxed{\Xi = [\xi_1 \xi_2 \cdots \xi_n]}. \tag{8}$$

Equation (7) may be taken as the definition of the discrete GFs, since it is clear that the jth GF simply describes the linear response of the system to the jth node's specified boundary value $\overline{\mathsf{v}}_j$. This equation may be interpreted as the discrete manifestation of a continuous GF integral equation; for example, a continuous representation may be written, in an obvious notation, as

$$v(\mathbf{x}) = \int_{\Gamma_u} \Xi_u(\mathbf{x}, \mathbf{y}) u(\mathbf{y}) d\Gamma_{\mathbf{y}} + \int_{\Gamma_p} \Xi_p(\mathbf{x}, \mathbf{y}) p(\mathbf{y}) d\Gamma_{\mathbf{y}}. \tag{9}$$

Once the GFs have been computed for one BVP type, that BVP may then be solved easily using (7). An attractive feature for interactive applications is that the entire n-vertex solution can be obtained in $18ns$ flops[1] if only s boundary values (BV) are nonzero (or have changed since the last timestep); moreover, fewer than n individual components of the solution may also be computed independently at proportionately smaller costs.

Parameterized body force contributions may in general also be included in (7) with an additional summation

$$\mathsf{v} = \Xi\overline{\mathsf{v}} + \mathsf{B}\beta, \tag{10}$$

[1]Flops convention [Golub and Loan 1996]: count both + and ×. For example, the scalar saxpy operation $y := a * x + y$ involves 2 flops, so that the 3-by-3 matrix-vector multiply accumulate, $\mathsf{v}_i := \Xi_{ij}\overline{\mathsf{v}}_j + \mathsf{v}_i$, involves 9 saxpy operations, or 18 flops.

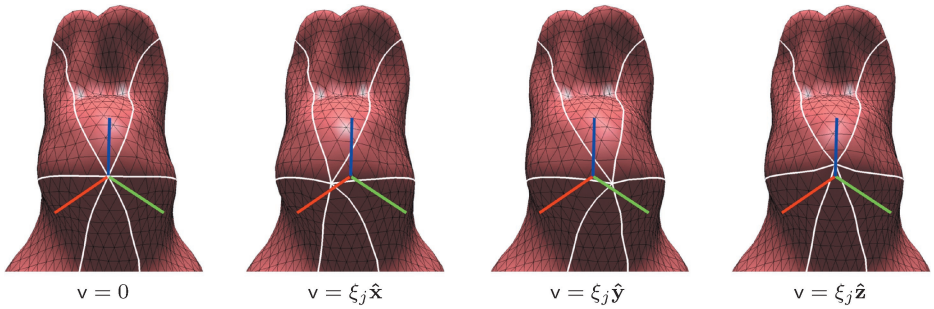

Fig. 3. Jth Green's function block column, $\xi_j = \Xi_{:j}$, representing the model's response due to the three XYZ components of the jth specified boundary value $\bar{\mathsf{v}}_j$. Here the vertex belongs to the ("free") traction boundary $j \in \Lambda_p$, and so ξ_j is literally the three responses due to unit tractions applied in the (RGB color-coded) XYZ directions. White edges emanating from the (displaced) jth vertex help indicate the resulting deformation. Note that the vertex does not necessarily move in the direction of the XYZ tractions. Using linear superposition, the CMA can determine the combinations of these and other tractions required to move vertices to specified positions.

where the sensitivity matrix B may be precomputed, and β are some scalar parameters. For example, gravitational body force contributions can be parameterized in terms of gravitational acceleration 3-vector $\mathbf{g}$.

Temporal coherence may also be exploited by considering the effect of individual changes in components of $\bar{\mathsf{v}}$ on the solution v. For example, given a sparse set of changes to the constraints $\delta\bar{\mathsf{v}}$, it follows from (7) that the new solution can be incremented efficiently,

$$\bar{\mathsf{v}}^{new} = \bar{\mathsf{v}}^{old} + \delta\bar{\mathsf{v}} \tag{11}$$

$$\mathsf{v}^{new} = \mathsf{v}^{old} + \Xi\,\delta\bar{\mathsf{v}}. \tag{12}$$

By only summing contributions to constraints that have changed significantly, temporal coherence can be exploited to reduce BVP solve times and obtain faster frame rates.

Further leveraging linear superposition, each precomputed GF system response may be enhanced with additional information for simulating other quantities such as volumetric stress, strain, and displacement data at selected locations. The multiresolution methods presented later can efficiently accomodate such extensions.

3.1.4 *Green's Function Precomputation.* It is important to realize that the GF models can have a variety of origins. Most obvious is numerical precomputation using standard tools such as the finite element [Zienkiewicz 1977] or boundary element methods [Brebbia et al. 1984]. In this case, the GF relationship between nodal variables in (6) and (7) provides a clear BVP definition for their computation (e.g., one GF scalar-column at a time). Reality-based scanning techniques provide a very different approach: empirical measurements of real physical objects may be used to estimate portions of the GF matrix for the scanned geometric model [Pai et al. 2001; Lang 2001]. Regardless of GF origins, the GF data abstraction nicely permits a variety of models to be used with this article's GF simulation algorithms. See Figure 3.

3.2 Fast Global Deformation Using Capacitance Matrix Algorithms

This section presents the Capacitance Matrix Algorithm for using the precomputed GFs of a relevant *reference BVP* (RBVP) type to efficiently solve other BVP types, and is foundational background material for this article.

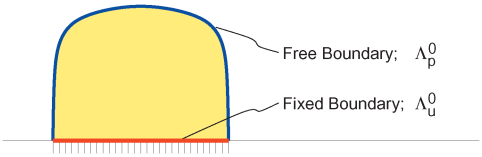

Fig. 4. Reference boundary value problem definition. The RBVP associated with a model attached to a flat rigid support is shown with boundary regions having displacement ("fixed," Λ_u^0) or traction ("free," Λ_p^0) nodal constraints indicated. A typical simulation would then impose contacts on the free boundary via displacement constraints with the capacitance matrix algorithm.

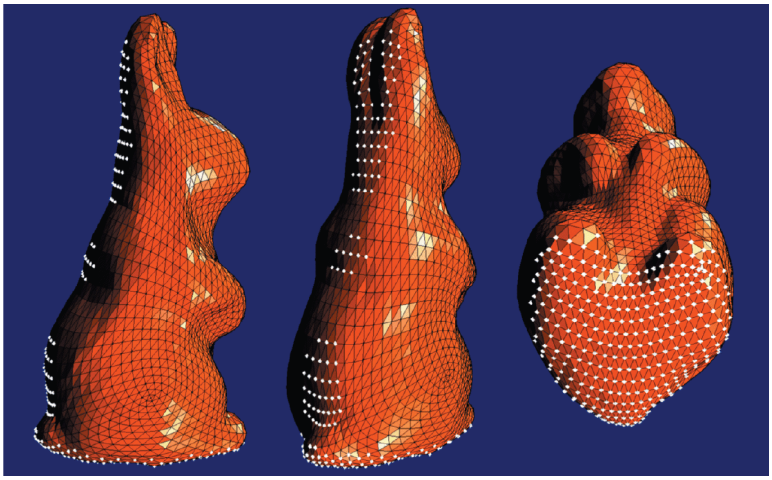

Fig. 5. Rabbit model reference boundary value problem. A RBVP for the rabbit model is illustrated with white dots attached to position constrained vertices in Λ_u^0. These (zero) displacement constraints were chosen to hold the rabbit model upright while users pushed on his belly in a force feedback simulation.

3.2.1 *Reference Boundary Value Problem Choice.* A key step in the precomputation process is the choice of a particular BVP type for which to precompute GFs. We refer to this as the *reference BVP*, and denote it by $(\Lambda_u^0, \Lambda_p^0)$, since its GFs are used in the CMA's updating process to solve all other BVP types encountered during a simulation. For interactions with an exposed free boundary, a common choice is to have the uncontacted model attached to a rigid support (see Figures 4 and 5). The GF matrix for the RBVP is hereafter referred to as Ξ.

3.2.2 *Capacitance Matrix Algorithm Formulae.* Precomputed GFs speed up the solution to the RBVP, but they can also dramatically reduce the amount of work required to solve a related BVP when used in conjunction with CMAs. If this were not so, precomputing GFs for a single BVP would have little practical use.

As motivation for changing BVP types, consider the very important case for force-feedback rendering where a manipulandum imposes contact displacement constraints (so that contact force output may be rendered) in a contact zone that has traction constraints in the RBVP. This new BVP type (with contact displacements instead of tractions) has different GFs than the RBVP, but the CMA can effectively solve the new BVP by determining the particular combination of contact tractions (and hence the linear combination of RBVP GFs) that satisfy the imposed displacement constraints.

Suppose the constraint-type changes (e.g., displacement↔traction), with respect to the RBVP at s nodes specified by the list of nodal indices are

$$\mathsf{S} = \{\mathsf{S}_1, \mathsf{S}_2, \ldots, \mathsf{S}_s\}. \tag{13}$$

The solution to this new BVP will then be

$$\mathsf{v} = \Xi^{new}\bar{\mathsf{v}} + \mathsf{B}^{new}\beta \tag{14}$$

for some "new" dense GF and body force matrices. Fortunately, using the Sherman–Morrison–Woodbury formula, the rank-3s modified GF and body force matrices may be written in the following useful factored form [James and Pai 1999, 2001],

$$\Xi^{new} = (\mathsf{I} + (\mathsf{E} + (\Xi\mathsf{E}))\mathsf{C}^{-1}\mathsf{E}^{\mathsf{T}})[\Xi(\mathsf{I} - \mathsf{E}\mathsf{E}^{\mathsf{T}}) - \mathsf{E}\mathsf{E}^{\mathsf{T}}] \tag{15}$$

$$\mathsf{B}^{new} = (\mathsf{I} + (\mathsf{E} + (\Xi\mathsf{E}))\mathsf{C}^{-1}\mathsf{E}^{\mathsf{T}})\mathsf{B}, \tag{16}$$

where E is an n-by-s block matrix

$$\mathsf{E} = \left[\mathsf{I}_{:\mathsf{S}_1}\mathsf{I}_{:\mathsf{S}_2}\cdots\mathsf{I}_{:\mathsf{S}_s}\right] \tag{17}$$

containing columns of the n-by-n identity block matrix I, specified by the list of updated nodal indices S. Postmultiplication[2] by E *extracts* columns specified by S. The resulting *capacitance matrix formulae* for v are

$$\boxed{\mathsf{v} = \underbrace{\mathsf{v}^{(0)}}_{n\times 1} + \underbrace{(\mathsf{E} + (\Xi\mathsf{E}))}_{n\times s}\underbrace{\mathsf{C}^{-1}}_{s\times s}\underbrace{\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)}}_{s\times 1}}, \tag{18}$$

where C is the s-by-s *capacitance matrix*, a negated submatrix of Ξ,

$$\boxed{\mathsf{C} = -\mathsf{E}^{\mathsf{T}}\Xi\mathsf{E}}, \tag{19}$$

and $\mathsf{v}^{(0)}$ is the response of the RBVP system to $\bar{\mathsf{v}}$,

$$\mathsf{v}^{(0)} = [\Xi(\mathsf{I} - \mathsf{E}\mathsf{E}^{\mathsf{T}}) - \mathsf{E}\mathsf{E}^{\mathsf{T}}]\bar{\mathsf{v}} + \mathsf{B}\beta. \tag{20}$$

3.2.3 *A Capacitance Matrix Algorithm for Global Solution.* With Ξ precomputed, formulae (18) through (20) immediately suggest an algorithm given that only simple manipulations of Ξ and inversion of the smaller capacitance submatrix are required. An algorithm for computing *all* components of v is as follows.

—For each new BVP type (with a different C matrix) encountered, construct and temporarily store C^{-1} (or LU factors) for subsequent use.

—Construct $\mathsf{v}^{(0)}$.

—Extract $\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)}$ and apply the capacitance matrix inverse to it, $\mathsf{C}^{-1}(\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)})$.

—Add the s column vectors $(\mathsf{E} + (\Xi\mathsf{E}))$ weighted by $\mathsf{C}^{-1}(\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)})$ to $\mathsf{v}^{(0)}$ for the final solution v.

Each new capacitance matrix inversion/factorization involves $\mathcal{O}(s^3)$ work, after which each solve takes no more than $\mathcal{O}(ns)$ operations given $\mathcal{O}(s)$ nonzero boundary values. This is particularly attractive when $s \ll n$ is small, such as often occurs in practice with localized surface contacts.

[2]Throughout, E is used to write sparse matrix operations using dense data, for example, Ξ, and like the identity matrix, it should be noted that there is no cost involved in multiplication by E or its transpose.

3.2.4 *Selective Deformation Computation.* A major benefit of the CMA with precomputed GFs is that it is possible to evaluate only selected components of the solution vector at run-time, with the total computing cost proportional to the number of components desired [James and Pai 2001]. This *random access* is a key enabling feature for haptics where contact force responses are desired at faster rates than the geometric deformations. The ability to exclusively simulate the model's response at desired locations is a very unique property of precomputed LEGFMs. Selective evaluation is also useful for optimizing (self) collision detection queries, as well as avoiding simulation of occluded or undesired portions of the model. We note that selective evaluation already provides a mechanism for multiresolution rendering of displacement fields generated using the CMA algorithm, however, this approach lacks the fast summation benefits that will be provided by wavelet GFs.

4. WAVELET GREEN'S FUNCTIONS

The Green's function based capacitance matrix algorithm has many appealing qualities for simulation, however, it does suffer inefficiencies when used for complex geometric models or large systems of updated constraints. Fortunately, these limitations mostly arise from using dense matrix representations for the discrete GF integral operator, and can be overcome by using multiresolution bases to control the amount of data that must be manipulated.

Displacement and traction fields associated with deformations arising from localized loads exhibit significant multiscale properties such as being very smooth in large areas away from the loading point (and other constraints) and achieving a local maxima about the point. For this reason, free space GFs (or fundamental solutions) of unbounded elastic media and our boundary GFs of 3D objects are efficiently represented by multiresolution bases. And just as this property of free space fundamental solutions allows for effective wavelet discretization methods for a wide range of integral equations [Beylkin et al. 1991a], it will also allow us to construct sparse wavelet representations of discrete elastostatic GF integral operators obtained from numerical solutions of constrained geometric models as well as measurements of real-world objects.

One could treat the GF matrix Ξ as a generic operator to be efficiently represented for full matrix-vector multiplication, as in standard wavelet approaches for fast iterative methods that involve transforms of both matrix rows and columns [Beylkin 1992], but this would be inefficient in our application on several grounds. One key reason is that column-based GF operations (such as weighted summations of selected GF columns) dominate the CMA's fast solution process. This is because solutions commonly exist in low-dimensional subspaces of the GF operator. Wavelet transforming all GF columns together destroys the ability of the CMA solver to efficiently compose these subspace solutions. Second, GF element extraction must be a relatively cheap operation in order for capacitance matrices to be obtained cheaply at run-time. Dense matrix formats allow elements to be extracted at O(1) cost, but wavelet transformed columns and/or rows introduce additional overhead. Because of such CMA GF usage concerns, *during precomputation we represent individual GF columns of the large GF matrix* Ξ *in the wavelet basis, but we do not transform along GF rows*.[3] Instead, row-based multiresolution constraints are addressed in Section 6. Requirements affecting the particular choice of wavelet scheme are discussed in Section 4.2.

4.1 Domain Structure of the Green's Function Matrix

Each GF column vector describes nodal traction and displacement distributions on different domains of the boundary, both of which have different smoothness characteristics. Interfaces between domains are

[3]Neglecting wavelet transforms of rows is not as bad as it may seem, partly because significant speedup is already obtained from transforming columns.

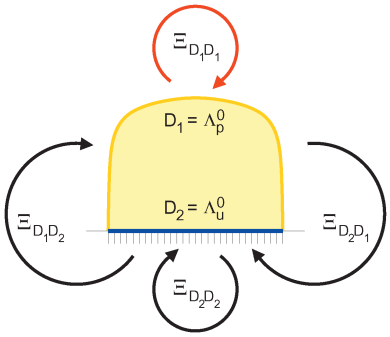

Fig. 6. Correspondence between boundary domain influences and domain block structure of the GF matrix Ξ: the influences between two boundary domains are illustrated here by arrows; each arrow represents the role of a GF block $\Xi_{D_iD_j}$ in the flow of information from specified BVs on domain D_j to unspecified BVs on domain D_i. For example, consider the *self-effect* of the exposed contactable surface (red arrow at top) which is of primary practical interest for deformation visualization. Each column of $\Xi_{D_1D_1}$ represents a displacement field on D_1 that decribes the effect of a traction applied (or displacement using the CMA) over some portion of D_1; this displacement field is efficiently represented using wavelets in Section 4.

therefore associated with discontinuities and the adjacent traction and displacement function values can have very different magnitudes and behaviors. For these reasons, multiresolution analysis of GFs is performed separately on each domain to achieve best results. From a practical standpoint, this also aids in simulating individual domains of the model independently (Section 3.2.4).

Domains are constructed by first partitioning nodes into Λ_u^0 and Λ_p^0 lists for which the GFs describe tractions and displacements, respectively. These lists are again split into disjoint subdomains if the particular wavelet transform employed cannot exploit coherence between these nodes. Let the boundary nodes be partitioned into d domains

$$D = \{D_1, \ldots, D_d\} \quad \text{with} \quad \bigcap_{i=1}^{d} D_i = \emptyset, \tag{21}$$

where D_i is a list of nodes in the natural coarse-to-fine resolution order of that domain's wavelet transform.

The d domains introduce a natural row and column ordering for the GF matrix Ξ which results in a clear block structure

$$\Xi = \sum_{i,j=1}^{d} \mathsf{E}_{D_i} \Xi_{D_iD_j} \mathsf{E}_{D_j}^{\mathsf{T}} \tag{22}$$

$$= \begin{bmatrix} \mathsf{E}_{D_1} \mathsf{E}_{D_2} \ldots \mathsf{E}_{D_d} \end{bmatrix} \begin{bmatrix} \Xi_{D_1D_1} & \Xi_{D_1D_2} & \cdots & \Xi_{D_1D_d} \\ \Xi_{D_2D_1} & \Xi_{D_2D_2} & & \vdots \\ \vdots & & \ddots & \\ \Xi_{D_dD_1} & \cdots & & \Xi_{D_dD_d} \end{bmatrix} \begin{bmatrix} \mathsf{E}_{D_1}^{\mathsf{T}} \\ \mathsf{E}_{D_2}^{\mathsf{T}} \\ \vdots \\ \mathsf{E}_{D_d}^{\mathsf{T}} \end{bmatrix}, \tag{23}$$

where the (i, j) GF block

$$\Xi_{D_iD_j} = \mathsf{E}_{D_i}^{\mathsf{T}} \Xi \mathsf{E}_{D_j} \tag{24}$$

maps data from domain D_i to D_j as illustrated in Figure 6.

4.2 Multiresolution Analysis and Fast Wavelet Transforms

By design, various custom multiresolution analyses and fast wavelet transforms can be used in the framework developed here provided they yield interactive inverse transform speeds (for fast summation), good GF compression (even for small models given that precomputation costs quickly increase), support for level of detail computations, ease of transform definition on user-specified surface domains D_i, and support for data from a wide range of discretizations.

As a result of these constraints we exploit work on biorthogonal lifted fast wavelet transforms based on second-generation wavelets derived from the lifting scheme of Sweldens and others [Sweldens 1998; Daubechies and Sweldens 1996; Schröder and Sweldens 1995a], and we refer the reader to those references for details; a related summary in the context of LEGFMs is available in James [2001]. We also consider Linear and Butterfly wavelets with a single lifting step; that is, the dual wavelet has one vanishing moment.

4.2.1 *Multiresolution Mesh Issues.* We use multiresolution triangle meshes with subdivision connectivity to conveniently define wavelets and MR constraints (Section 6) as well as provide detailed graphical and haptic rendering (Section 9). Many of our meshes have been modeled using Loop subdivision [Loop 1987] which trivializes the generation of multiresolution meshes. General meshes may be reparameterized using approaches from the literature [Eck et al. 1995; Krishnamurthy and Levoy 1996; Lee et al. 1998; Guskov et al. 2000; Lee et al. 2000] and also commercially available packages [Raindrop Geomagic, Inc.: Paraform]. For our purposes, we implemented algorithms based on normal meshes [Guskov et al. 2000; Lee et al. 2000], and have used the related displaced subdivision surface approach [Lee et al. 2000] for rendering detailed deforming models. Examples of models we have reparameterized are the rabbit model (Figures 1 and 12; original mesh courtesy of Cyberware, and the dragon model (Figure 8; original mesh courtesy of the Stanford Computer Graphics Laboratory). For the dragon model some undesirable parameterization artifacts are present, however, this can be avoided with more care.

For good wavelet compression results, it is desirable to have many subdivision levels for a given model. This also aids in reducing the size of the dense base level GF data, if they are left unthresholded. In cases where the coarsest resolution of the mesh is still large, reparameterization should be considered but it is still possible to consider more exotic lifted wavelets on arbitrary point sets. To maximize the number of levels for modest models (e.g., for the rabbit model), we resorted to manual fitting of coarse base-level parameterizations, although more sophisticated approaches are available [Eck et al. 1995; Krishnamurthy and Levoy 1996; Lee et al. 1998; Guskov et al. 2000]. Although this is clearly a multiresolution mesh generation issue, how to design meshes that optimize wavelet GF compression (or other properties) is a nonobvious open problem. Finally, adaptive meshing must be used with care since coarse regions limit the ability to resolve surface deformations and constraints.

4.2.2 *Multilevel Vertex Notation.* For the semiregular meshes, we denote mesh levels by $0, 1, 2, \ldots, L$ where 0 is the coarse base level, and the finest level is L. All vertices on a level l are denoted by the index set $\mathcal{K}(l)$, so that all mesh vertices are contained in $\mathcal{K}(L) = \{1, 2, \ldots, n\}$. These index sets are nested to describe the multiresolution structure, so that level $l + 1$ vertices $\mathcal{K}(l + 1)$ are the union of "even" vertices $\mathcal{K}(l)$ and "odd" vertices $\mathcal{M}(l)$ so that

$$\mathcal{K}(l + 1) = \mathcal{K}(l) \cup \mathcal{M}(l), \tag{25}$$

and this is illustrated in Figure 7. Consequently, the vertex domain sets D also inherit a multiresolution structure.

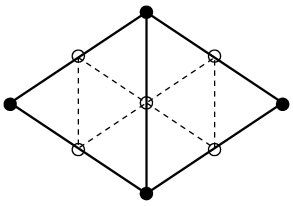

Fig. 7. Multilevel vertex sets: a simple two-level surface mesh patch on level $j+1$ (here $j=0$). The four *even* vertices (solid dots) belong to the base mesh and constitute $\mathcal{K}(j)$, whereas the *odd* vertices of $\mathcal{M}(j)$ all correspond to edge-splits ("midpoints") of parent edges. The union of the two sets is the set of all vertices on level $j+1$; namely, $\mathcal{K}(j+1) = \mathcal{K}(j) \cup \mathcal{M}(j)$.

4.3 Wavelet Transforms on Surface Domains

Consider the forward and inverse fast wavelet transform (FWT) pair $(\mathsf{W}, \mathsf{W}^{-1})$, itself composed of FWT pairs

$$\mathsf{W} = \sum_{i=1}^{d} \mathsf{E}_{D_i} \mathsf{W}_i \mathsf{E}_{D_i}^{\mathsf{T}} = \mathsf{E}_D \left(\mathrm{diag}_i(\mathsf{W}_i)\right) \mathsf{E}_D^{\mathsf{T}} \tag{26}$$

$$\mathsf{W}^{-1} = \sum_{i=1}^{d} \mathsf{E}_{D_i} \mathsf{W}_i^{-1} \mathsf{E}_{D_i}^{\mathsf{T}} = \mathsf{E}_D \left(\mathrm{diag}_i\left(\mathsf{W}_i^{-1}\right)\right) \mathsf{E}_D^{\mathsf{T}} \tag{27}$$

with the ith pair $(\mathsf{W}_i, \mathsf{W}_i^{-1})$ defined on domain D_i. For brevity we refer the reader to Sweldens [1998] and Schröder and Sweldens [1995a] for background on the implementation of lifted Linear and Butterfly wavelet transforms; the details of our approach to adapting the lifted transforms to vertex domains D are described in James [2001] (see §3.1.5).

4.4 Wavelet Green's Functions

Putting things together, the wavelet transform of the GF matrix is then

$$\mathsf{W}\Xi = [(\mathsf{W}\xi_1)(\mathsf{W}\xi_2)\cdots(\mathsf{W}\xi_n)] = \sum_{i,j=1}^{d} \mathsf{E}_{D_i} \left(\mathsf{W}_i \, \Xi_{D_i D_j}\right) \mathsf{E}_{D_j}^{\mathsf{T}} \tag{28}$$

or with a shorthand "tilde" notation for transformed quantities,

$$\tilde{\Xi} = [\tilde{\xi}_1 \, \tilde{\xi}_2 \cdots \tilde{\xi}_n] = \sum_{i,j=1}^{d} \mathsf{E}_{D_i} \left(\tilde{\Xi}_{D_i D_j}\right) \mathsf{E}_{D_j}^{\mathsf{T}}. \tag{29}$$

The individual block component of the jth *wavelet GF* $\tilde{\xi}_j = \tilde{\Xi}_{:j}$ corresponding to vertex i on level l of domain d is denoted with rounded bracket subscripts as

$$(\tilde{\xi}_j)_{(l,i;d)} = \tilde{\Xi}_{(l,i;d)j}. \tag{30}$$

This notation is complicated but no more so than necessary, since it corresponds directly to the multiresolution data structure used for implementation.

4.5 Tensor Wavelet Thresholding

Each 3-by-3 block of the GF matrix describes a tensor influence between two nodes. The wavelet transform of a GF (whose row elements are 3×3 matrix blocks) is mathematically equivalent to nine scalar transforms, one for each tensor component. However, in order to reduce run-time sparse matrix overhead (and improve cache hits), we evaluate all transforms at the block level. For this reason, our

thresholding operation either accepts or rejects an entire block. Whether performing the transforms at the scalar component level improves matters, despite increasing the sparse indexing storage and run-time overhead up to a factor of nine, is a subject of future work.

Our oracle for wavelet GF thresholding compares the Frobenius norm of each block wavelet coefficient[4] to a domain– and level-specific thresholding tolerance, and sets the coefficient to zero if it is smaller. Thresholding of the jth wavelet GF, $\tilde{\xi}_j$, on a domain d is performed for the ith coefficient if and only if $i \in D_d$, $i \in \mathcal{M}(l)$, and

$$\left\| \tilde{\Xi}_{ij} \right\|_F < \varepsilon_l \left\| \mathsf{E}_{D_d}^{\mathsf{T}} \xi_j \right\|_{\infty F}, \tag{31}$$

where

$$\left\| \mathsf{E}_{D_d}^{\mathsf{T}} \xi_j \right\|_{\infty F} \equiv \max_{i \in D_d} \| \Xi_{ij} \|_F \tag{32}$$

is a weighted measure of GF amplitude on domain d, and ε_l is a level-dependent relative threshold parameter decreased on coarser levels (smaller l) as

$$\varepsilon_l = 2^{l-L}\varepsilon, \quad l = 1, \ldots, L, \tag{33}$$

with ε the user-specified threshold parameter. We usually do not threshold base level ($l = 0$) coefficients even when this introduces acceptable errors because the lack of response, such as, pixel motion, in these regions can be perceptually bothersome.

For our models we have observed stable reconstruction of thresholded data; for example,

$$\left\| \mathsf{E}_{D_d}^{\mathsf{T}} \left(\xi_j - \mathsf{W}^{-1} \tilde{\xi}_j \right) \right\|_{\infty F} < C\varepsilon \left\| \mathsf{E}_{D_d}^{\mathsf{T}} \xi_j \right\|_{\infty F} \tag{34}$$

typically for some constant C near 1. Examples are shown in Section 10. Although there are no guarantees that wavelet bases constructed on any particular model will form an unconditional basis, and so the thresholding operation will lead to stable reconstructions, none of our numerical experiments with discrete GFs have suggested anything to the contrary. Similar experiences were reported by the pioneers of the lifting scheme in Schröder and Sweldens [1995a] for wavelets on the sphere. Some formal conditions on the stability of multiscale transformations are proven in Dahmen [1996]. Results illustrating the relationship between error and thresholding tolerance are presented later (in Section 10).

4.6 Storage and Transmission of Green's Functions

Wavelets provide bases for sparsely representing GFs, but further compression is possible for storage and transmission data formats. We note that efficient wavelet quantization and coding schemes [DeVore et al. 1992; Shapiro 1993; Said and Pearlman 1996] have been extended to dramatically reduce file sizes of surface functions compressed using the lifting scheme [Kolarov and Lynch 1997], and similar approaches can be applied to GF data.

5. CMA WITH FAST SUMMATION OF WAVELET GFS

The CMA is slightly more involved when the GFs are represented in wavelet bases. The chief benefit is the performance improvement obtained by using the FWT for fast summation of GF and body force responses.

[4]The Frobenius norm of a real-valued 3-by-3 matrix a is

$$\|a\|_F = \sqrt{\sum_{ij} a_{ij}^2}.$$

5.1 Motivation

In addition to reducing memory usage, it is well known that by sparsely representing our GF columns in a wavelet basis we can use the FWT for fast matrix multiplication [Beylkin et al. 1991a]. For example, consider the central task of computing a weighted summation of s GFs,

$$\sum_{j \in \mathsf{S}} \xi_j \bar{\mathsf{v}}_j, \tag{35}$$

involving sn 3×3 matrix-vector multiply-accumulate operations. Quick evaluation of such expressions is crucial for fast BVP solution (c.f. (7)) and graphical rendering of deformations, and it is required at least once by the CMA solver. Unfortunately, as s increases this operation quickly becomes more and more costly and as $s \to n$ eventually involves $O(n^2)$ operations. By using a FWT it is possible to perform such sums more efficiently in a space in which the GF columns are approximated with sparse representations.

The weighted GF summation can be rewritten by premultiplying (35) with the identity operator $\mathsf{W}^{-1}\mathsf{W}$:

$$\sum_{j \in \mathsf{S}} \xi_j \bar{\mathsf{v}}_j = \mathsf{W}^{-1} \sum_{j \in \mathsf{S}} \tilde{\xi}_j \bar{\mathsf{v}}_j. \tag{36}$$

By precomputing sparse thresholded approximations of the wavelet transformed GFs, $\tilde{\Xi}$, a fast summation will result in (36) provided that the advantage of sparsely representing Ξ more than compensates for the extra cost of applying W^{-1} to the vector data. This occurs in practice, due to the FWT's speed and excellent decorrelation properties for GF data.

5.2 Formulae

The necessary formulae result from substituting

$$\Xi = \mathsf{W}^{-1}\mathsf{W}\Xi \tag{37}$$

into the CMA formulae (18) through (20), and using the GF expression (29). The result may be written as

$$\mathsf{v} = \mathsf{v}^{(0)} + (\mathsf{E} + \mathsf{W}^{-1}(\tilde{\Xi}\mathsf{E}))\mathsf{C}^{-1}\left(\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)}\right) \tag{38}$$

$$\mathsf{C} = -(\mathsf{E}^{\mathsf{T}}\mathsf{W}^{-1})(\tilde{\Xi}\mathsf{E}) \tag{39}$$

$$\mathsf{v}^{(0)} = \mathsf{W}^{-1}[\tilde{\Xi}(\mathsf{I} - \mathsf{E}\mathsf{E}^{\mathsf{T}})\bar{\mathsf{v}} + \tilde{\mathsf{B}}\beta] - \mathsf{E}\mathsf{E}^{\mathsf{T}}\bar{\mathsf{v}} \tag{40}$$

$$\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)} = (\mathsf{E}^{\mathsf{T}}\mathsf{W}^{-1})[\tilde{\Xi}(\mathsf{I} - \mathsf{E}\mathsf{E}^{\mathsf{T}})\bar{\mathsf{v}} + \tilde{\mathsf{B}}\beta] - \mathsf{E}^{\mathsf{T}}\bar{\mathsf{v}}, \tag{41}$$

where we have taken the liberty of sparsely representing the parameterized body force contributions in the wavelet basis. With these formulae, it is possible to evaluate the solution v using only one inverse FWT evaluation and some partial reconstructions $\mathsf{E}^{\mathsf{T}}\mathsf{W}^{-1}$.

5.3 Selective Wavelet Reconstruction Operation

The operator $(\mathsf{E}^{\mathsf{T}}\mathsf{W}^{-1})$ represents the reconstruction of a wavelet transformed function at the updated nodes S. This is required in at most two places: capacitance matrix element extraction from $\tilde{\Xi}$; and evaluation of $(\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)})$ in cases when the first term of $\mathsf{v}^{(0)}$ (in square brackets) is nonzero. It follows from the tree structure of the wavelet transform that these extraction operations can be evaluated efficiently with worst-case per-element cost proportional to the logarithm of the domain size. Although such approaches were sufficient for our purposes, in practice several optimizations related to spatial and temporal data structure coherence can significantly reduce this cost. For example, portions of C

are usually cached and so extraction costs are amortized over time, typically with very few entries required per new BVP. Also, spatial clustering of updated nodes leads to the expected cost of extracting several clustered elements being not much more than the cost of extracting one. Furthermore, spatial clustering in the presence of temporal coherence allows us to exploit coherence in a sparse GF wavelet reconstruction tree, so that nodes that are topologically adjacent in the mesh can expect to have elements reconstructed at very small costs. For these reasons, it is possible to extract capacitance matrix entries at a fraction of the cost of LU factorization. Performance results for block extraction operations are given in Section 10.5. The logarithmic cost penalty introduced by wavelet representations is further reduced in the presence of hierarchical constraints, and a hierarchical variant of the fast summation CMA is discussed in Section 8.

5.4 Algorithm

An efficient algorithm for computing the entire solution vector v is possible by carefully evaluating subexpressions in the convoluted manner:

(1) Given constraints, $\bar{\mathsf{v}}$, and the list of nodes to be updated, S,

(2) Obtain C^{-1} (or factorization) for this BVP type from the cache (*Cost*: Free), using updating (see James [2001]), or from scratch (*Cost:* $2s^3/3$ flops);

(3) If nonzero, evaluate the sparse summation

$$\tilde{\mathsf{g}}_1 = [\tilde{\Xi}(\mathsf{I} - \mathsf{E}\mathsf{E}^{\mathsf{T}})\bar{\mathsf{v}} + \tilde{\mathsf{B}}\beta]. \tag{42}$$

(*Cost:* $18\bar{s}\tilde{n}$ flops from first term, where $\tilde{n}$ is the average number of nonzero 3-by-3 blocks per wavelet GF being summed (in practice $\tilde{n} \ll n$), and $\bar{s}$ is the number of nonupdated nonzero constraints. The second body force term is similar but ignored due to ambiguity. Cost can be reduced by exploiting temporal coherence, e.g., see (12).)

(4) Compute the block s-vector

$$\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)} = (\mathsf{E}^{\mathsf{T}}\mathsf{W}^{-1})\tilde{\mathsf{g}}_1 - \mathsf{E}^{\mathsf{T}}\bar{\mathsf{v}}. \tag{43}$$

(*Cost:* Selective reconstruction cost (if nontrivial g_1) $3sR_{\mathsf{S}}$, where R_{S} is the effective cost of reconstructing a scalar given S (discussed in Section 5.3; expected cost is $R_{\mathsf{S}} = \mathcal{O}(1)$, worst case cost is $R_{\mathsf{S}} = \mathcal{O}(\log n)$), plus $3s$ flops for addition.)

(5) Evaluate the block s-vector

$$\mathsf{g}_2 = \mathsf{C}^{-1}\big(\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)}\big). \tag{44}$$

(*Cost:* $18s^2$ flops.)

(6) Perform the sparse summation

$$\tilde{\mathsf{g}}_1 \mathrel{+}= (\tilde{\Xi}\mathsf{E})\mathsf{g}_2. \tag{45}$$

(*Cost:* $18s\tilde{n}$ flops.)

(7) Perform inverse FWT (can be performed in place on block 3-vector data)

$$\mathsf{v} = \mathsf{W}^{-1}\tilde{\mathsf{g}}_1. \tag{46}$$

(*Cost:* $3C_{\mathsf{IFWT}}n$ flops, where C_{IFWT} is approximately four for lifted Linear wavelets.)

(8) Correct updated values to obtain the final solution

$$\mathsf{v} \mathrel{+}= \mathsf{E}(\mathsf{g}_2 - \mathsf{E}^{\mathsf{T}}\bar{\mathsf{v}}). \tag{47}$$

(*Cost:* $6s$ flops.)

5.5 Cost Analysis

The total cost of evaluating the solution is

$$Cost = 3C_{\text{IFWT}}n + 18(s+\bar{s})\tilde{n} + 18s^2 + 3s(\mathsf{R}_\mathsf{S}+3) \quad \text{flops}, \tag{48}$$

where the notable improvement introduced by fast summation is the replacement of the $18sn$ dense summation cost with that of the sparse summation and inverse FWT. This excludes the cost of capacitance matrix inverse construction (or factorization or updating), if updating is performed, since this is experienced only once per BVP type and amortized over frames.

Two interesting special cases are when nonzero constraints are either all updated ($\bar{s}=0$) or when no constraints are updated ($s=0$). In the case where all nonzero constraints are updated ($\bar{s}=0$), and therefore Step 3 has zero g_1, the total cost of the calculation is

$$Cost = 3C_{\text{IFWT}}n + 18s\tilde{n} + 18s^2 + 3s(\mathsf{R}_\mathsf{S}+3) \quad \text{flops}. \tag{49}$$

Cases in which updated nodes have zero constraints are slightly cheaper. When no constraints are updated ($s=0$) only GF fast summation is involved, and the cost is

$$Cost = 3C_{\text{IFWT}}n + 18\bar{s}\tilde{n} \quad \text{flops}. \tag{50}$$

In practice we have reduced these costs by only reconstructing the solution on subdomains (reduces FWT cost and summation cost) where it is required, for example, for graphical rendering. It clearly follows that it is possible to reconstruct the solution at coarser resolutions for multiple LOD rendering, that is, by only evaluating g_1 and the IFWT in Step 7 for coarse resolutions, and this issue is discussed futher in Section 9.

We found this algorithm to be very effective for interactive applications, and especially for force feedback simulation with pointlike contacts (small $\bar{s}$ and $s=0$). Timings and typical flop counts are provided in the results, Section 10. For large models with many updated constraints, the $s\tilde{n}$ and s^2 contributions, in addition to the capacitance matrix inversion, can become costly. This issue is addressed in the following section by introducing multiresolution constraints that can favorably reduce the effective size of s.

6. HIERARCHICAL CONSTRAINTS

The MR GF representations make it feasible to store and simulate geometrically complex elastic models by eliminating the dominant bottlenecks associated with dense GF matrices. However, finer discretizations can introduce complications for real-time simulations that impose numerous constraints on these same fine scales: (1) even sparse fast summation will eventually become too costly as more GF columns contribute to the sum, and (2) updating numerous constraints with the CMA incurs costly capacitance matrix inversion costs.

We provide a practical solution to this problem that can also optionally reduce precomputation costs. Our approach is to reduce the number of constraints by imposing constraints at a coarser resolution than the geometric model (see Figure 8). This eliminates the aforementioned bottlenecks without sacrificing model complexity. Combined with wavelet GFs that enable true multiresolution BVP simulation and solution output, multiresolution constraints provide the BVP's complementary multiresolution input control. Such an approach is well suited to the CMA which effectively works by updating constraints defined over finite areas; in the continuous limit, as $n \to \infty$ and scaling function measures go to zero, the area affected by the uniresolution finite-rank-updating CMA also goes to zero and the CMA would have no effect.

The multiresolution constraints are described by nested spaces with node interpolating basis functions defined on each domain. Using interpolating scaling functions allows hierarchical constraints to

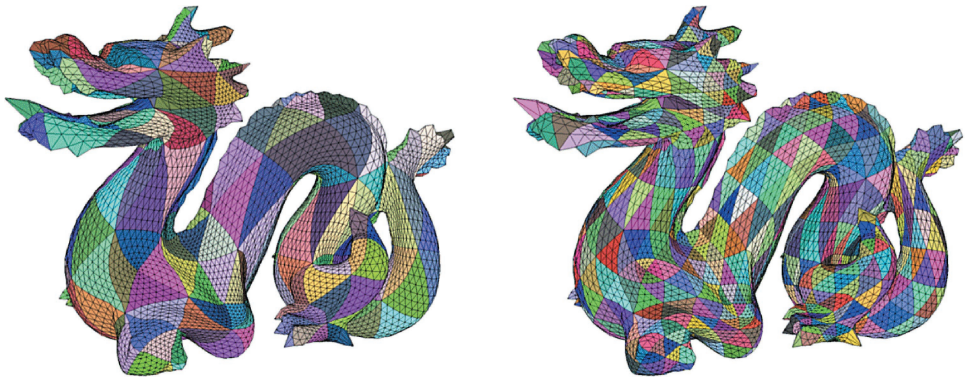

Fig. 8. Multiresolution constraint parameterizations: two dragon meshes ($L = 3$) with coarser constraint parameterizations indicated for different resolutions of the Green's function hierarchy; (left) constraints on level 0, and (right) on level 1. In this way, interactive traction constraints can be applied on the coarse scale while deformations are rendered using fine-scale displacement fields. (Reparameterized dragon model generated from mesh courtesy of Stanford Computer Graphics Laboratory.)

coexist with nodal constraint descriptions, which is useful for defining the hierarchical version of the CMA (in Section 8). For our piecewise linear function spaces these scaling functions correspond to *hierarchical basis functions*[5] [Yserentant 1986] and the interpolation filters are already available from the unlifted portion of the linear FWT used for the MR GFs.

Let the scalar hierarchical basis function

$$\phi_{[l,k;d]} = \phi_{[l,k;d]}(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \tag{51}$$

correspond to vertex index k belonging to level l and domain D_d. Here the square subscript bracket is used to indicate a hierarchical basis function; recall (Equation (30)) that rounded subscript brackets are used to refer to row components of wavelet transformed vectors or matrix columns. In this notation, the traditional "hat functions" on the finest scale are

$$\phi_k(\mathbf{x}) = \phi_{[L,k;d]}(\mathbf{x}), \quad k \in D_d. \tag{52}$$

In bracket notation, the refinement relation satisfied by these interpolating scaling functions is

$$\phi_{[l,k;d]} = \sum_{j \in \mathcal{K}(l+1)} h_{[l,k,j;d]}\, \phi_{[l+1,j;d]}, \tag{53}$$

where h is the (unlifted) interpolating refinement filter. As a result, the surface hierarchical basis functions are unit normalized

$$\phi_{[l,i;d]}(\mathbf{x}_{(l,j;d)}) = \delta_{ij}, \tag{54}$$

where δ_{ij} is the Kronecker delta function. The refinement relation for hierarchical basis functions implies that hierarchical constraint boundary values on finer constraint scales are given by interpolating subdivision, and so satisfy the refinement relation

$$\overline{\mathsf{v}}_{[l,:;:]} = \mathsf{H}_l^{\mathsf{T}} \overline{\mathsf{v}}_{[l+1,:;:]}, \tag{55}$$

[5]In a slight abuse of terminology, hereafter we collectively use "hierarchical basis functions" to denote the interpolating vertex-based hierarchical scaling functions even if the function space is not piecewise linear, such as Butterfly.

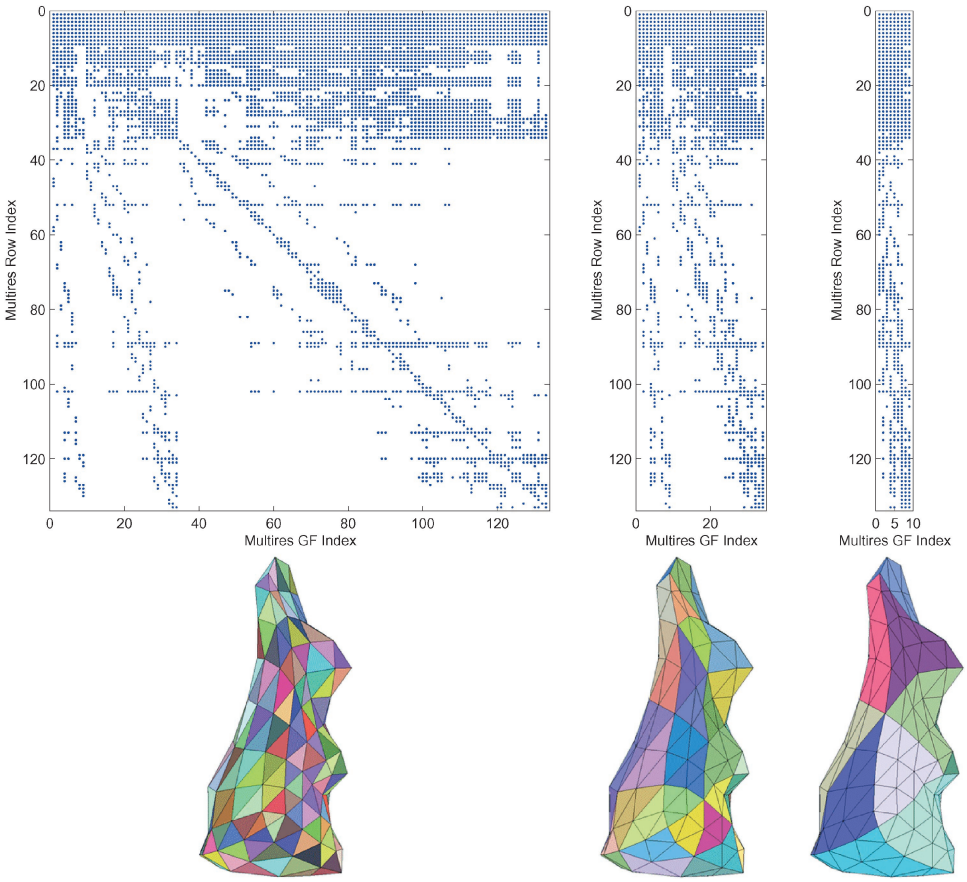

Fig. 9. Hierarchical wavelet GF matrix structure: sparsity patterns and constraint parameterizations of the coarse level 2 ($L = 2$) rabbit model's three-level GF hierarchy for the main Λ_p^0 "free-boundary" self-effect block $\Xi_{\Lambda_p^0\Lambda_p^0}$ (illustrated in Figure 6). This model has 160 vertices, with the lifted linear FWT defined on a domain of 133 vertices partitioned into three levels with sizes (9, 25, 99). The matrices are: (left) finest scale GF square matrix block (# nonzero blocks, $nnz = 4444$), (middle) once-coarsened constraint scale GF block ($nnz = 1599$), (right) twice-coarsened constraint scale GF block ($nnz = 671$). In each case, sparsity resulting from thresholding the wavelet transformed GF columns clearly illustrates the wavelet transform's excellent decorrelation ability. The multiresolution structure of the wavelet coefficients is apparent in each matrix as a result of multiresolution reordering of rows and columns; notice the dense unthresholded base-level coefficients in the topmost rows. Perhaps surprising for such a small model, modest compression ratios are already being obtained: here $\varepsilon = 0.10$ and the large block has retained $nnz = 4444$ elements or 25% of the original size.

where we have used a brief operator notation (equivalent to (53) except it relates 3-vector elements instead of scalars), or simply

$$\overline{\mathsf{v}}_{[l]} = \mathsf{H}_l^{\mathsf{T}} \overline{\mathsf{v}}_{[l+1]}. \tag{56}$$

As we now show, although the hierarchical constraints are described at a coarse resolution, the corresponding deformation response involves all scales.

7. HIERARCHICAL GREEN'S FUNCTIONS

The GF responses corresponding to each hierarchical constraint basis function are named *hierarchical GFs*. From a GF matrix perspective, the coarsening of the constraint scales is associated with

a reduction in GF columns (see Figure 9). A graphical illustration of hierarchical GFs is given in Figure 13.

7.1 Notation

The hierarchical GFs are identified using the square bracket notation introduced for HBFs: let

$$\xi_{[l,k;d]} = \Xi_{:,[l,k;d]} \tag{57}$$

denote the hierarchical GF associated with the kth vertex contained on level l and domain D_d. Therefore

$$\xi_{[0,k;d]}, \xi_{[1,k;d]}, \ldots, \xi_{[L,k;d]} \tag{58}$$

are all hierarchical GFs associated with the kth vertex here contained on the base level of the subdivision connectivity mesh. The hierarchical *wavelet* GFs (illustrated in Figure 9) are easily identified by both a tilde and square brackets; for example,

$$\tilde{\xi}_{[l,k;d]} = \tilde{\Xi}_{:,[l,k;d]}. \tag{59}$$

7.2 Refinement Relation

Hierarchical GFs and hierarchical basis functions share the same refinement filters since each hierarchical GF is expressed in terms of a linear combination of GFs on finer levels by

$$\xi_{[l,k;d]} = \sum_{j \in \mathcal{K}(l+1)} h_{[l,k,j;d]}\, \xi_{[l+1,j;d]} \tag{60}$$

or in operator notation

$$\Xi^l = \Xi^{l+1} \mathsf{H}_l^{\mathsf{T}}. \tag{61}$$

This follows from the hierarchical GF ansatz

$$\Xi^l \overline{\mathsf{v}}_{[l]} = \Xi^{l+1} \overline{\mathsf{v}}_{[l+1]}, \tag{62}$$

for a level l hierarchical constraint $\overline{\mathsf{v}}_{[l]}$, after substituting the hierarchical boundary condition subdivision equation (56),

$$\overline{\mathsf{v}}_{[l]} = \mathsf{H}_l^{\mathsf{T}} \overline{\mathsf{v}}_{[l+1]}. \tag{63}$$

Figure 9 provides intuitive pictures of the induced GF hierarchy,

$$\xi_{[L,*;d]}, \ldots, \xi_{[1,*;d]}, \xi_{[0,*;d]}. \tag{64}$$

7.3 Matrix BVP Definition

Although the refinement relation (61) can be used to compute coarse scale hierarchical GFs from finer resolutions, it is also possible to compute them directly using the definition of the accompanying hierarchical boundary value constraints. For example, the three columns of the hierarchical GF $\xi_{[l,k;d]}$ can be computed using a black-box solver (e.g., FEM), by solving three BVPs corresponding to ith vertex scalar constraint $\phi_{[l,k;d]}(\mathbf{x}_i)$ separately specified for x, y, and z components (with other components set to zero; analogous to Figure 3). This provides an attractive approach to hierarchically precomputing very large models, and was used for the large dragon model.

8. HIERARCHICAL CMA

It is possible to use the hierarchical GFs to produce variants of the CMA from Section 3.2. The key benefits obtained from using hierarchical GFs are related to the smaller number of constraints (see

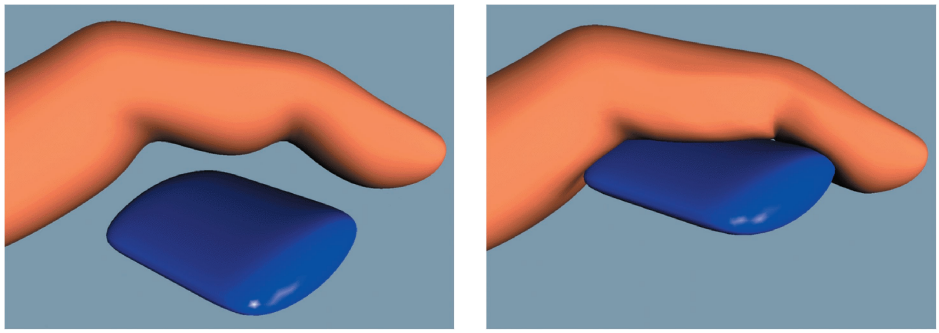

Fig. 10. A finger pad in contact with a flat surface is a good example of where hierarchical GFs are beneficial, as is any case where numerous dense surface constraints occur. Although the traction field may contain little information (e.g., smooth or nearly constant), large run-time costs can result from the number of GFs being summed and/or by the number of constraints being updated with a CMA. Whether the deformation is computed with the finger pad's free boundary constraints modeled by the user specifying tractions directly, or indirectly using displacements and a CMA, in both cases hierarchical GFs result in smaller boundable run-time costs.

Figure 11): (1) an accelerated fast summation (since fewer weighted columns need be summed), (2) smaller capacitance matrices, and (3) improved feasibility of caching potential capacitance matrix elements at coarse scales. Due to the four-fold change in vertex count per resolution level, the expected impact of reducing the constraint resolution by J levels is

(1) 4^J reduction in constraint count and number of GFs required in CMA summations,
(2) 16^J reduction in number of capacitance matrix elements,
(3) 64^J reduction in cost of factoring or directly inverting capacitance matrix, and
(4) $4^J - 64^J$ reduction in CMA cost.

An illustration of a situation where the hierarchical CMA can be beneficial is given in Figure 10.

It is relatively straightforward to construct a nonadaptive hierarchical CMA that simply limits updated displacement constraints to fixed levels of resolution. This is the easiest mechanism for providing graceful degradation when large sets of nodes require updating: if too many constraints are being too densely applied they may simply be resolved on a coarser scale. This is analogous to using a coarser level model, with the exception that the solution, for example, displacements, is available at a finer scale. We have found this simple approach works well in practice for maintaining interactivity during otherwise intensive updating cases. One drawback of the nonadaptive approach is that it can lead to "popping" when changing between constraint: resolutions, and the investigation of adaptive CMA variants for which this problem is reduced is work for the future.

8.1 Hierarchical Capacitances

Similar to the nonhierarchical case, hierarchical capacitance matrices are submatrices of the hierarchical GFs. We can generalize the capacitance node list definition to include updated nodal constraints corresponding to hierarchical basis functions at different resolutions. We first generalize the notation of the original (fine scale) capacitance node list and capacitance matrix elements as

$$\mathsf{S} = (k_1, k_2, \ldots, k_s) \tag{65}$$

$$= ([L, k_1; d_1], [L, k_2; d_2], \ldots, [L, k_s; d_s]) \tag{66}$$

$$\mathsf{C}_{ij} = -\Xi_{k_i[L,k_j;d_j]}. \tag{67}$$

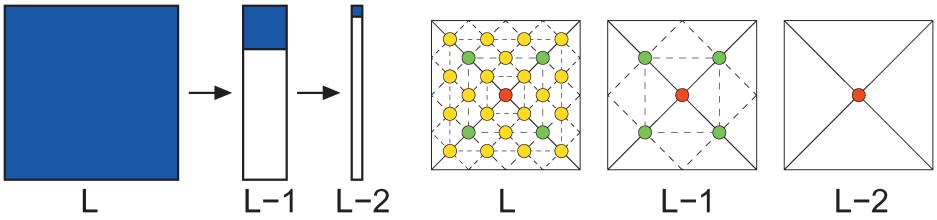

Fig. 11. (left) As in Figure 9, the matrix view of hierarchical GF indicates an approximately fourfold reduction in columns at each coarser constraint resolution. As a result, the number of possible capacitance matrix elements are reduced accordingly, as represented by the blue matrix blocks. (right) An illustration of the corresponding spatial hierarchy for the support of a coarse level (extraordinary) "linear hat" scaling function. Circles indicate the vertex nodes (and basis functions) required to represent the coarse level scaling function at each level.

Hierarchical constraints then follow by replacing L with the appropriate level. The CMA corresponding to coarsened constraint scales follows immediately, as well as the fact that hierarchical capacitance matrix inverses can be updated to add and delete hierarchical constraints. Furthermore, it is also possible to mix constraint scales and construct true multiresolution updates using the generalized definition

$$\mathsf{S} = ([l_1, k_1; d_1], [l_2, k_2; d_2], \ldots, [l_s, k_s; d_s]) \tag{68}$$

$$\mathsf{C}_{ij} = -\Xi_{k_i[l_j,k_j;d_j]}. \tag{69}$$

Such adaptivity can reduce the number of constraints required, which in turn reduces both the number of GFs summed, and the size of the capacitance matrix. However, due to the additional complexity of specifying adaptive multiresolution constraints at run-time (e.g., for an interactive contact mechanics problem), we have not yet exploited this CMA solver functionality in practice. Finally, due to the reduced number of constraints, there are fewer and smaller capacitance matrices, and this improves the effectiveness of caching strategies (see Figure 11).

8.2 Graceful Degradation

For real-time applications, hierarchical capacitances play an important role for resolving constraints on coarser constraint scales (or adaptively in general). Consider a simulation with constraints resolved on level H. If it encounters a capacitance matrix inverse update task that requires too much time it can abort and resort to resolving the problem at a coarser constraint resolution, for example, $H - 1$ or lower. In this way it is possible to find a coarse enough level at which things can proceed quickly.

As with all variants of the CMA, these direct matrix solution algorithms provide predictable operation counts, which may be used to choose an effective real-time solution strategy.

9. DETAILED GRAPHICAL AND HAPTIC RENDERING

At some scale, there is little practical benefit in seeking higher resolution elastic models, and geometric detail can be introduced by local mapping.

9.1 LOD and Multiresolution Displacement Fields

The fast summation CMA with wavelet GFs (Section 5) immediately provides an obvious mechanism for real-time adaptive level-of-detail (LOD) rendering [Xia et al. 1997]. This process is slightly complicated by the fact that the geometry is deforming, thereby reducing dependence on statically determined geometric quantities, such as, visibility. We have not explored real-time LOD in our implementation,

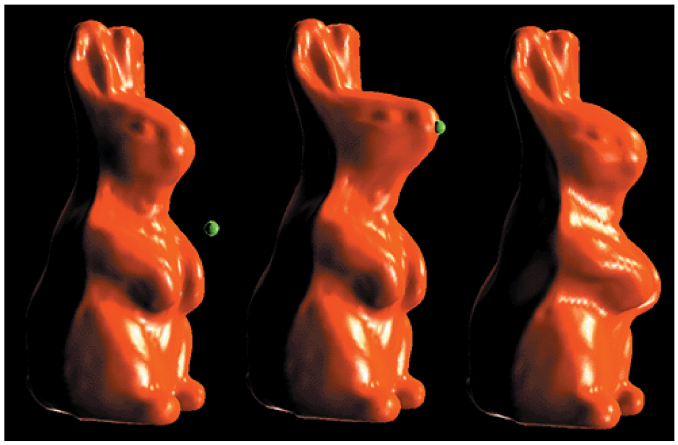

Fig. 12. Displaced subdivision surfaces provide a natural extension to a hierarchy of elastic spatial scales. In this example, a level-2 elastic rabbit model is rendered on level 5 using displacement mapping (computed in software). In addition to providing exact displacement constraints on detailed (or just subdivided) surfaces, hierarchical GFs allow greater elastic content to be depicted than simple displacement mapping of coarse geometry. In either case, such approaches can effectively transfer the run-time simulation burden almost entirely to graphical rendering.

however, it was an important algorithm design consideration. It also provides an extra mechanism for real-time graceful degradation for difficult CMA constraint problems.

9.2 Hierachical GFs and Geometric Detail

A favorable exploitation of spatial scales is obtained by using hierarchical GFs, since interactions resolved on relatively coarse constraints scales naturally allow visualization of fine scale geometry and displacement fields. Even when coarse level constraints are used, finer scale displacement fields are still available, possibly computed from a highly accurate discretization.

There is, however, an interesting transition at some spatial scale for which the GF displacement fields contain little more information than is obtained by displacement mapping a geometrically coarser resolution model. By evaluating GF displacement fields only to a suitable detail level, the deformed geometry can then be mapped to finer scales via bump and/or displacement mapping, possibly in graphics hardware. We have used displaced subdivision surfaces (DSS) [Lee et al. 2000] to illustrate this, partly because they work well with deforming meshes.

One significant concern when displacement mapping coarse models is that it can lead to inexact displacement constraints. This problem is exaggerated by DSS even for small changes due to mapping, because the Loop subdivision step converts our interpolating constraint scaling functions into noninterpolating ones. Intuitively, this occurs because adjacent vertex displacements computed by CMA for the coarse control mesh are averaged during the subdivision process, thus leading to inexact constraint values. This is in contrast to the interpolating constraints achieved with hierarchical GFs. Nevertheless, for finely meshed models the mismatch caused by displacement mapping is reduced.

One setting for which we have found DSS to be still very useful is for haptic force feedback applications involving pointlike contacts. Here perceptual problems related to surface penetration due to inaccurate surface displacement constraints are commonly overcome by a “god-object approach” [Zilles and Salisbury 1994] in which a proxy for the object in contact with the surface is always drawn on the surface (the “god” object) regardless of whether penetration occurs. We have successfully used this in several pointlike contact interactive force feedback simulations; for example, see Figure 12.

Table I. Properties of Rabbit and Dragon Models Used in Multiresolution Experiments[a]

Model	Tetra	Face	Vertex, n	|Domain|	$\|\mathcal{M}(l)\|$	MB
Rabbit 2	872	320	162	133	(9, 25, 99)	.64
Rabbit 3	6903	1280	642	537	(9, 26, 101, 401)	10
Rabbit 4	54475	5120	2562	2145	(9, 26, 100, 404, 1606)	166
Dragon 3	176702	19840	9920	7953	(123, 372, 1495, 5963)	2277

[a]Columns are provided for the number of triangles and vertices on the boundary, an estimate of the number of tetrahedra for a uniform tetrahedralization, and the size of the Λ_p^0 domain along with its partitioned level structure on which the wavelet GFs are analyzed. For comparison, the last column indicates the memory size (in MB) of the otherwise uncompressed dense $\Xi_{\Lambda_p^0\Lambda_p^0}$ matrix of 32-bit floats.

Table II. Green's Function Precomputation and Simulation Times for Rabbit BEM Models[a]

Model	Tetra	Face	Vertex,n	|Domain|	Precomp	Sim (ms)
Rabbit 2	872	320	162	133	1.8 min	0.07
Rabbit 3	6903	1280	642	537	40 min	0.33
Rabbit 4	54475	5120	2562	2145	9 hours[b]	1.3

[a]Only GFs corresponding to movable free vertices (in Λ_p^0) were precomputed, and representative times are listed (Precomp). The last column indicates that (sub)millisecond graphics-loop computations (Sim) are required to determine the pointlike contact deformation response of each model's free boundary (e.g., for force feedback simulations).
[b]Note: The rabbit 4 model was precomputed on an 8-way PIII-450 MHz machine.

9.3 Force Feedback Rendering of Detail

In addition to graphical rendering, surface detail may also enhance force feedback rendering by using normal maps to modulate point contact friction forces [Morgenbesser and Srinivasan 1996] as is done in commercial force feedback systems, for example, Reachin. In this way, the hierarchical GFs parameterize the coarse scale force response of the compliant surface, and the normal maps render surface detail.

10. RESULTS

In addition to the images and examples already presented, results presented here illustrate the effectiveness of methods presented for wavelet GF compression, fast summation, and hierarchical techniques. An accompanying video illustrates our models used in a force-feedback simulation.

All multiresolution analysis is performed on the Λ_p^0 domain, and GF compression is concerned with the $\Xi_{\Lambda_p^0\Lambda_p^0}$ GF self-effect block, since it is of greatest practical importance in simulations. As a reminder, this GF block describes surface displacments on Λ_p^0 due to tractions applied to Λ_p^0. Several models have been analyzed and are described in Table I. A fair estimate[6] of the number of tetrahedra in corresponding uniform tetrahedralizations are also stated.

10.1 Note on Java Timings

Timings are reported later for precomputation (Table II), element extraction (Table III), and fast summation (Figure 17). All timings were computed on a Pentium III-450 MHz machine with 256 MB, running Windows 98 and the Sun JDK 1.2.2 JVM. Based on the average performance of the representative 3-by-3 blocked matrix-vector multiplication (18*537 flop in 0.19 ms) *this Java computing*

[6] Tetrahedra counts are based on dividing the volume of the model V by the volume of a regular tetrahedron V_{tet} with triangle face area equal to the mesh's mean face area a:

$$V_{tet} = \frac{192^{1/4}}{9} a^{3/2} \Rightarrow \#\text{Tetrahedra} \approx \left\lceil \frac{V}{0.4126 a^{3/2}} \right\rceil.$$

Table III. Pessimistic Timings of Selective Reconstruction Operations for GF 3-by-3 Block Element Extraction[a]

# Levels	2	3	4
Time/block, μsec	8	20	36

[a]Block extraction times are listed as a function of the number of resolution levels (# Levels) that must be adaptively reconstructed to obtain the element.

environment is approximately rated at a modest 51 MFlops. Significantly better performance (tenfold improvement) is possible using current hardware with optimized linear algebra libraries.

10.2 Wavelet GF Compression and Error Examples

This section shows that substantial GF compression can be obtained at the cost of introducing very practical levels of approximation error. The practical consequence is that *specifying the level of simulation error allows the speedup of our interactive simulations to be directly controlled*, and this is extremely useful for real-time applications.

10.2.1 *Measures of Error.* For a given level of compression, we give two measures of the error in the reconstructed GF matrix block $\hat{\Xi}_{\Lambda_p^0\Lambda_p^0}$ relative to the exact value $\Xi_{\Lambda_p^0\Lambda_p^0}$. The first error estimate is based on the relative Frobenius (or Euclidean) norm of the error, here called the "RMS" error,

$$RMS = \frac{\left\| \hat{\Xi}_{\Lambda_p^0\Lambda_p^0} - \Xi_{\Lambda_p^0\Lambda_p^0} \right\|_F}{\left\| \Xi_{\Lambda_p^0\Lambda_p^0} \right\|_F}, \tag{70}$$

and is a robust estimate of the average GF matrix element error. The second estimate provides a measure of the maximum relative blockwise error over all GFs, here called the "MAX" error,

$$MAX = \max_{j \in \Lambda_p^0} \frac{\left\| \mathsf{E}_{\Lambda_p^0}^{\mathsf{T}} (\hat{\xi}_j - \xi_j) \right\|_{\infty F}}{\left\| \mathsf{E}_{\Lambda_p^0}^{\mathsf{T}} \xi_j \right\|_{\infty F}}, \tag{71}$$

where $\|\cdot\|_{\infty F}$ is defined in Equation (32).

10.2.2 *Rabbit Model.* Compression results for the five ($L = 4$) level rabbit model of Figure 1 are shown in Figure 14. An image of the compressed GF matrix for the smaller $L = 2$ rabbit model was also shown earlier in Figure 9. In general, the results indicate a trend toward greater compression ratios for larger models (this is characterized further in Section 10.3). In order to illustrate the performance benefit of lifting the Linear and Butterfly wavelets, results obtained using the unlifted bases are also shown for reference. To avoid clutter in our plots, the generally less effective unlifted wavelet results are plotted in a lighter color for clarity. Graphical depiction of the errors associated with GF compression are shown in Figure 13. Some representative precomputation times for the rabbit BEM models are shown in Table II.

The relationship of relative RMS and MAX errors to the relative thresholding tolerance ε for various wavelets is shown in Figure 15 ($L = 4$). Interestingly, the behavior of errors for Linear and Butterfly wavelets is nearly identical for respective lifted and unlifted types.

10.3 Dependence of GF Compression on Model Complexity

To better understand how compression or fast summation speedup rates depend on the Λ_p^0 domain resolution of a model, the ratio of fast summation speedup factors for models of adjacent resolutions $(L+1)$ and L are shown in Figure 16 as a function of relative RMS error. Given a model with m vertices in its Λ_p^0 domain, the fast summation speedup factor is defined as the ratio of the number of dense GF

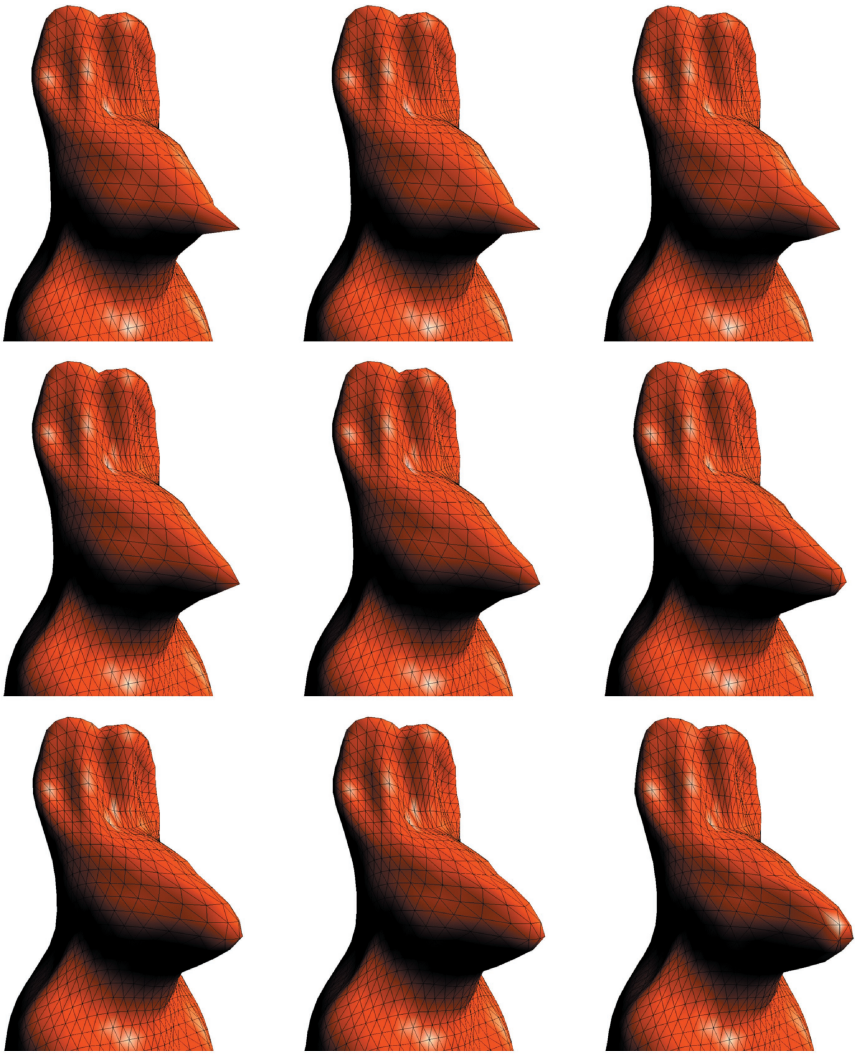

Fig. 13. Rabbit model ($L = 4$) approximate wavelet GF reconstructions for lifted Linear wavelets at three thresholds, $\varepsilon = (0.01, 0.05, 0.20)$, corresponding to compression factors of (8.4, 25, 68). Three hierarchical GFs are shown with constraint levels 2 (top row), 3 (middle row), and 4 (bottom row), and were computed using the refinement relation from fine scale (level-4) thresholded GFs. Relative errors proportional to the threshold are visible, especially in the neighborhood of the rabbit's nose where an exaggerated normal displacement constraint has been applied to each model.

elements m^2 to the number of nonzero wavelet GF blocks nnz, or

$$speedup(m) = \frac{m^2}{nnz(m, \varepsilon)}. \tag{72}$$

The ratio of speedup for two adjacent levels with m and $4m$ vertices is therefore

$$\frac{speedup(4m)}{speedup(m)} = \frac{(4m)^2}{nnz(4m, \varepsilon)} \frac{nnz(m, \varepsilon)}{m^2} = \frac{16 nnz(m, \varepsilon)}{nnz(4m, \varepsilon)}. \tag{73}$$

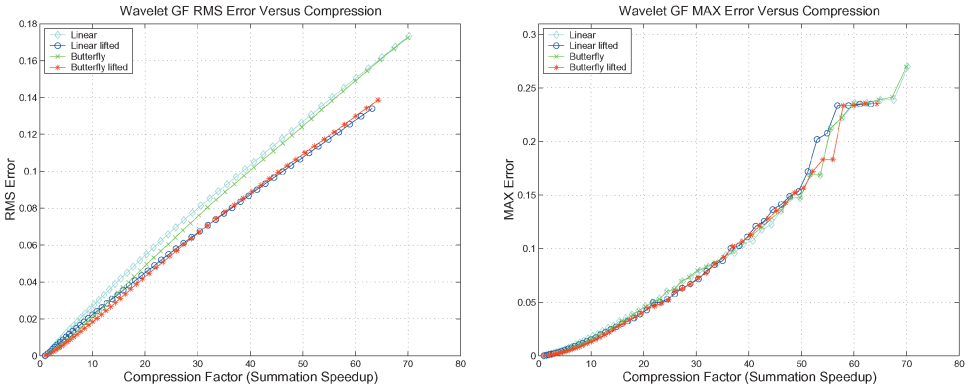

Fig. 14. Rabbit model ($L = 4$): Wavelet GF error versus compression.

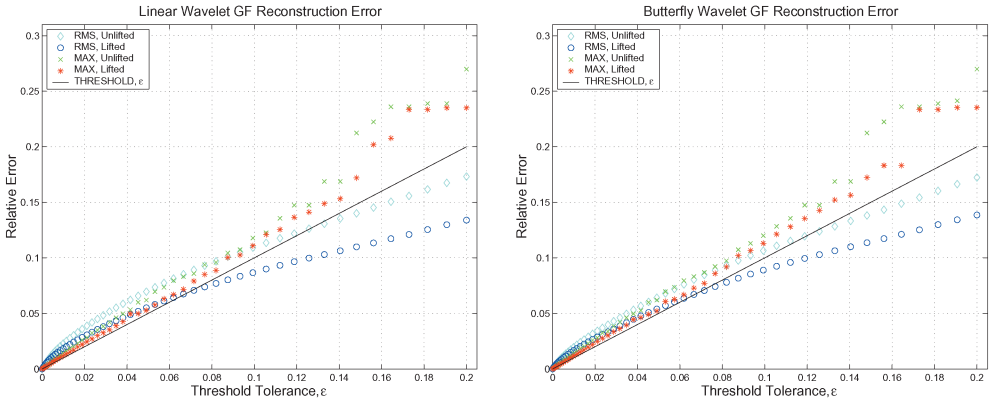

Fig. 15. Rabbit model ($L = 4$)—Wavelet GF error versus thresholding tolerance: (left) Linear wavelets; (right) Butterfly wavelets.

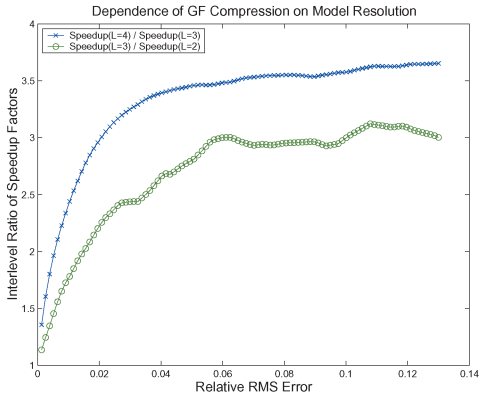

Fig. 16. Rabbit model: dependence of GF compression on model resolution.

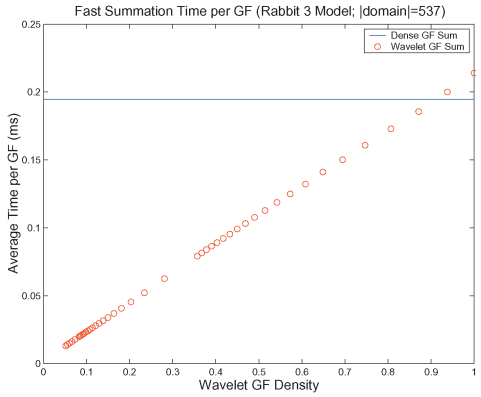

Fig. 17. Comparison of wavelet GF fast summation timings (in milliseconds) of a rabbit model ($L = 3$, 537 vertex domain) with dense GF matrix multiplication (horizontal line, time = 0.19 ms/GF) for full matrix multiplication. The linear dependence on nonzero GF matrix elements confirms the cost analysis of Section 5.5 (Equation (50): fast summation costs are directly proportional to the number of nonzero wavelet GF elements. Timings are for the lifted Linear wavelets, for which the inverse FWT requires 0.38 ms.

To provide intuition, linear dependence of the number of nonzeros $nnz(m, \varepsilon)$ on m would yield a ratio of 4; whereas for $nnz(m, \varepsilon) = C_\varepsilon m \log^\alpha m$, $\alpha \geq 0$, one would obtain

$$\frac{speedup(4m)}{speedup(m)} = \frac{4 \log^\alpha(m)}{\log^\alpha(4m)} \leq 4. \tag{74}$$

Although the limited information in Figure 16 does not allow us to confidently estimate the exact dependence of nnz on m, it does provide a very useful observation regarding the dependence of the ratio of fast summation speedups on error: in practice there is little improvement in relative speedup between resolutions once the RMS error level has increased to a certain level.

10.4 Verification of Fast Summation Speedup

Fast summation speedups are directly related to the compression achieved using wavelets. Experimental evidence for the linear dependence of fast summation speedup on GF compression is illustrated in Figure 17. Our run-time simulations experienced close to a proportional speedup, with the inverse lifted Linear wavelet transform being approximately as costly as an extra normal computation. We do not use Butterfly wavelets for our interactive simulation because the negligible compression benefits (if any) do not outweigh the increased cost of the inverse wavelet transform.[7] As the number of constraints increases and GF response summations dominate the graphics simulation cost, *speedups from wavelet fast summation directly translate into speedups for interactive simulations.*

10.5 Timings of Selective Wavelet Reconstruction Operations ($E^T W^{-1}$)

The performance of inverse FWT operations for the extraction of GF block elements (Section 5.3) are shown in Table III for an unoptimized recursive element reconstruction implementation using Linear wavelets. The implementation's reconstruction of each element involves redundant calculation overhead of approximately a factor of two. Nevertheless, these pessimistic times are sufficiently fast for practical use and can be optimized further using the approaches mentioned in Section 5.3.

[7]Butterfly subdivision requires averaging of four times as many values as Linear, however, it can be efficiently implemented to be only twice as costly.

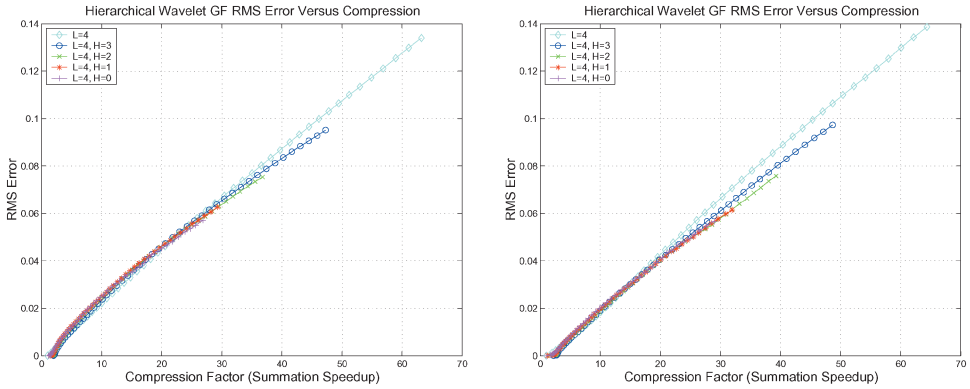

Fig. 18. Rabbit model ($L = 4$). Hierarchical wavelet GF error versus compression: (left) Lifted Linear; (right) Lifted Butterfly.

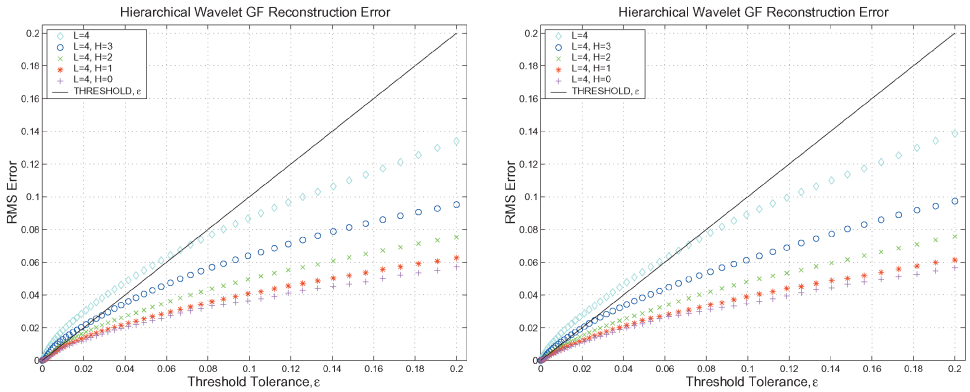

Fig. 19. Rabbit model ($L = 4$). Hierarchical wavelet GF error versus thresholding tolerance: (left) Lifted Linear; (right) Lifted Butterfly.

10.6 Wavelet Compression of Hierarchical Green's Functions

Wavelet compression results are shown in Figure 18 for hierarchical GFs corresponding to the rabbit model. Compression behaviors for each level of the GF hierarchy are approximately the same, although the coarser and therefore smoother GF levels result in only slightly better compression for a given RMS error, with this being more apparent for the smoother lifted Butterfly basis. Figure 19 displays RMS reconstruction error versus thresholding tolerance for each level of the hierarchy. The approximately equivalent compression rates for GFs across constraint scales imply that a fourfold reduction in constraints per coarsened constraint level results in approximately a fourfold speedup in fast summation for a given level of error.

The four-level dragon ($L = 3$) is our largest model, with 19,840 faces, 9920 vertices, and 7953 vertices in the Λ_p^0 domain partitioned across four levels of sizes $(123, 372, 1495, 5963)$. In order to reduce the precomputation time, we only computed hierarchical GFs at the coarsest ($l = 0$) constraint scale (illustrated in Figure 8), resulting in 123 GFs precomputed instead of 7953. The deformations associated with these coarse level constraints are very smooth (shown in Figure 22), and for this reason the good compression is achieved (see Figures 20 and 21).

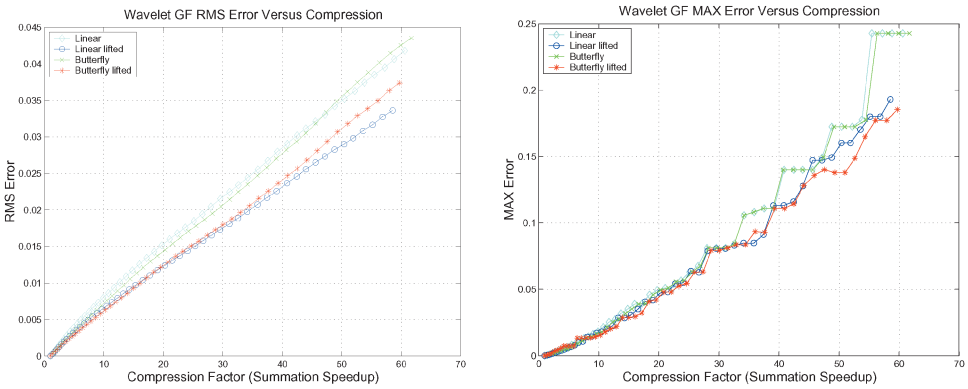

Fig. 20. Dragon model ($L = 3$). Hierarchical wavelet GF error versus compression.

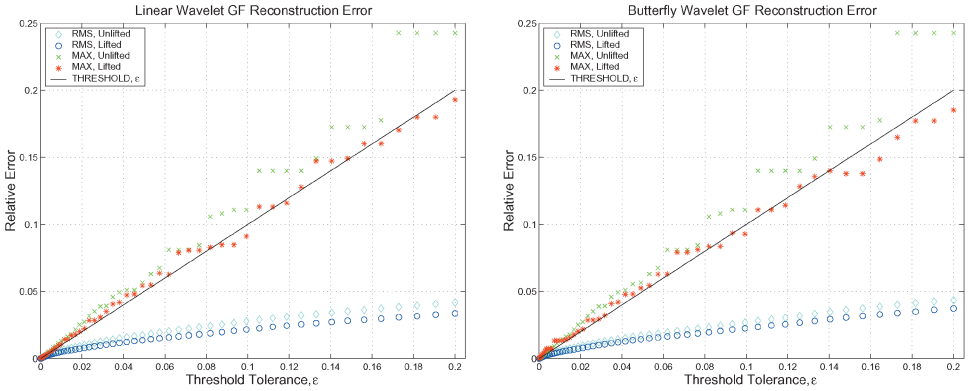

Fig. 21. Dragon model ($L = 3$). Hierarchical wavelet GF error versus thresholding tolerance: (left) Linear wavelets; (right) Butterfly wavelets.

11. SUMMARY AND CONCLUSIONS

This article has outlined a multiresolution framework for interactive simulation of large-scale Green's function based physical models which is a significant improvement over previously known approaches. Our contribution builds on Capacitance Matrix Algorithm approaches for simulating precomputed GF models associated with discrete approximations of linear elliptic partial differential equations, such as for elastostatic objects. The numerous multiresolution enhancements presented here (hierarchical wavelet GFs, fast summation CMA, multiresolution constraints) offer dramatic improvements in the overall effectiveness of the CMA, and greatly extend the complexity of models that can be interactively simulated and also precomputed. In particular, the fast summation simulation enhancements that exploit wavelet GF compression were highly successful: we have shown that hundredfold reductions in interactive simulation costs for geometrically complex elastic models (dragon $L = 3$) can be achieved at acceptable levels of error (5% RMS), and that similar compression rates are in fact possible for a wide range of wavelet schemes. With these improvements in efficiency and extensions for large-scale simulation, interactive GF-based models will be of much greater use in interactive computer graphics, computer haptics, and related applications such as virtual prototyping, video games, character animation, surgical simulation, and interactive assembly planning.

Fig. 22. Deformed dragon model ($L = 3$) shown in a force feedback simulation, contacted by a small white ball (force direction indicated by arrow). The undeformed model is shown in the top left image, and other images illustrate hierarchical GF deformations due to forces resolved on constraint level 0. This very large model compresses extremely well (approx. factor of 100 at 5% RMS error) and is therefore suitable for interactive force feedback simulation.

11.1 Future Work

There are a number of promising directions for future research with perhaps the largest area being the simulation and visualization of other physical systems not directly related to elastostatics, for example, thermostatics, hydrostatics, electrostatics, and so on. For elasticity, we are also investigating extensions for geometric and material nonlinearities that can still exploit aspects of the fast precomputed GF simulation approach.

The benefit of other wavelet schemes should be studied for compression improvements and practical concerns such as the accommodation of common discretizations and definitions on irregular (base) meshes. Other issues related to smoothness of discretization spaces, larger models, and high-accuracy tolerances should also be considered. Recently it has been shown that smooth wavelets based on Loop subdivision can achieve excellent compression ratios for complicated geometries [Khodakovsky et al. 2000] at the cost of an expensive (but here affordable) forward transform, and these schemes would have desirable properties here for compression, summation, and visual smoothness, as well as storage and transmission (the central motivation in Khodakovsky et al. [2000]). This is an avenue for future research, however, we note that our results do not initially suggest that such wavelets will provide significant improvement.

The compression of GF matrix blocks other than the free surface self-effect (illustrated in Figure 6, should be investigated; preliminary studies indicate that this is also effective. Algorithms for adaptive multiresolution approximations of contact constraints for real-time simulation are needed. Collision detection for deformable objects can be optimized for LEGFMs given the efficient access to state values and sensitivities. Issues related to the stable simulation of models that contain errors need to be better understood; this is centrally related to the simulation of wavelet compressed models and also models acquired with physical measurement. Lastly, the methods presented here are suitable for hard real-time simulation environments and could be further studied in such a context.

REFERENCES

ALPERT, B., BEYLKIN, G., COIFMAN, R., AND ROKHLIN, V. 1993. Wavelet-like bases for the fast solution of second-kind integral equations. *SIAM J. Sci. Comput. 14*, 1 (Jan.), 159–184.

ASCHER, U. M. AND PETZOLD, L. R. 1998. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, Philadelphia.

ASTLEY, O. AND HAYWARD, V. 1998. Multirate haptic simulation achieved by coupling finite element meshes through Norton equivalents. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

BALANIUK, R. 2000. Building a haptic interface based on a buffer model. In *Proceedings of the IEEE International Conference on Robotics and Automation* (San Francisco).

BARAFF, D. AND WITKIN, A. 1992. Dynamic simulation of non-penetrating flexible bodies. In *Computer Graphics (SIGGRAPH 92 Conference Proceedings)*, 303–308.

BARAFF, D. AND WITKIN, A. 1998. Large steps in cloth simulation. In *SIGGRAPH 98 Conference Proceedings*, 43–54.

BARBER, J. R. 1992. *Elasticity*, first ed. Kluwer, Dordrecht.

BASDOGAN, C. 2001. Real-time simulation of dynamically deformable finite element models using modal analysis and spectral Lanczos decomposition methods. In *Proceedings of the Medicine Meets Virtual Reality (MMVR'2001) Conference,* 46–52.

BERKLEY, J., WEGHORST, S., GLADSTONE, H., RAUGI, G., BERG, D., AND GANTER, M. 1999. Fast finite element modeling for surgical simulation. In *Proceedings of Medicine Meets Virtual Reality*, 55–61.

BEYLKIN, G. 1992. On the representation of operators in bases of compactly supported wavelets. *SIAM J. Numer. Anal. 29*, 6 (Dec.), 1716–1740.

BEYLKIN, G., COIFMAN, R., AND ROKHLIN, V. 1991a. Fast wavelet transforms and numerical algorithms. *Commun. Pure Appl. Math. 44*, 141–183.

BEYLKIN, G., COIFMAN, R., AND ROKHLIN, V. 1991b. Fast wavelet transforms and numerical algorithms I. *Commun. Pure Appl. Math. 44*, 141–183.

BREBBIA, C. A., TELLES, J. C. F., AND WROBEL, L. C. 1984. *Boundary Element Techniques: Theory and Applications in Engineering*, second ed. Springer-Verlag, New York.

BRO-NIELSEN, M. AND COTIN, S. 1996. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. *Comput. Graph. Forum 15*, 3 (Aug.), 57–66.

ÇAVUŞOĞLU, M. C. AND TENDICK, F. 2000. Multirate simulation for high fidelity haptic interaction with deformable objects in virtual environments. In *Proceedings of the IEEE International Conference on Robotics and Automation* (San Francisco).

COTIN, S., DELINGETTE, H., AND AYACHE, N. 1999. Realtime elastic deformations of soft tissues for surgery simulation. *IEEE Trans. Vis. Comput. Graph. 5*, 1, 62–73.

CYBERWARE. Available at http://www.cyberware.com.

DAHMEN, W. 1996. Stability of multiscale transformations. *J. Fourier Anal. Appl. 4*, 341–362.

DAUBECHIES, I. AND SWELDENS, W. 1996. Factoring wavelet transforms into lifting steps. Tech. Rep. Bell Laboratories, Lucent Technologies.

DEBUNNE, G., DESBRUN, M., BARR, A., AND CANI, M.-P. 2001. Dynamic real-time deformations using space and time adaptive sampling. In *Computer Graphics (SIGGRAPH 2001 Conference Proceedings)*.

DESBRUN, M., SCHRÖDER, P., AND BARR, A. 1999. Interactive animation of structured deformable objects. In *Proceedings of Graphics Interface*, 1–8.

DeVore, R., Jawerth, B., and Lucier, B. 1992. Image compression through wavelet transform coding. *IEEE Trans. Inf. Theor. 38*, 719–746.

Dyn, N., Levin, D., and Gregory, J. A. 1990. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Trans. Graph. 9*, 2 (Apr.), 160–169.

Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., and Stuetzle, W. 1995. Multiresolution analysis of arbitrary meshes. *Comput. Graph. 29*, Annual Conference Series, 173–182.

Ezawa, Y. and Okamoto, N. 1989. High-speed boundary element contact stress analysis using a super computer. In *Proceedings of the Fourth International Conference on Boundary Element Technology*, 405–416.

Gibson, S. F. and Mirtich, B. 1997. A survey of deformable models in computer graphics. Tech. Rep. TR-97-19, Mitsubishi Electric Research Laboratories, Cambridge, Mass. November.

Golub, G. H. and Loan, C. F. V. 1996. *Matrix Computations*, third ed. Johns Hopkins University Press, Baltimore.

Gortler, S. J., Schröder, P., Cohen, M. F., and Hanrahan, P. 1993. Wavelet radiosity. In *Computer Graphics Proceedings, Annual Conference Series*, 221–230.

Greengard, L. and Rokhlin, V. 1987. A fast algorithm for particle simulations. *J. Comput. Phys. 73*, 325–348.

Guskov, I., Vidimce, K., Sweldens, W., and Schröder, P. 2000. Normal meshes. In *SIGGRAPH 2000 Conference Proceedings*, K. Akeley, Ed., Annual Conference Series. ACM Press/ACM SIGGRAPH/Addison Wesley Longman, Reading, Mass., 95–102.

Hackbusch, W. 1985. *Multi-Grid Methods and Applications*. Springer, Berlin.

Hackbusch, W. and Nowak, Z. P. 1989. On the fast matrix multiplication in the boundary element method by panel clustering. *Numerische Mathematik 54*, 4, 463–491.

Hager, W. W. 1989. Updating the inverse of a matrix. *SIAM Rev. 31*, 2 (June), 221–239.

Hartmann, F. 1985. *The mathematical foundation of structural mechanics*. Springer-Verlag, New York.

Hauth, M. and Etzmuss, O. 2001. A high performance solver for the animation of deformable objects using advanced numerical methods. In *Proceedings of Eurographics 2001* (to appear).

Hirota, K. and Kaneko, T. 1998. Representation of soft objects in virtual environments. In *Proceedings of the Second International Conference on Artificial Reality and Tele-Existence*.

James, D. L. 2001. Multiresolution Green's function methods for interactive simulation of large-scale elastostatic objects and other physical systems in equilibrium. Ph.D. Thesis, Institute of Applied Mathematics, University of British Columbia, Vancouver, British Columbia, Canada.

James, D. L. and Pai, D. K. 1999. ArtDefo: Accurate real time deformable objects. *Comput. Graph. 33*, Annual Conference Series, 65–72.

James, D. L. and Pai, D. K. 2001. A unified treatment of elastostatic contact simulation for real time haptics. *Haptics-e, Elect. J. Haptics Res. 2,* 1 (Sept.). Available at www.haptics-e.org.

James, D. L. and Pai, D. K. 2002a. DyRT: Dynamic response textures for real time deformation simulation with graphics hardware. In *SIGGRAPH 2002 Conference Proceedings*. Annual Conference Series. ACM Press/ACM SIGGRAPH (to appear).

James, D. L. and Pai, D. K. 2002b. Real time simulation of multizone elastokinematic models. In *Proceedings of the IEEE International Conference on Robotics and Automation* (Washington, DC), 927–932.

Jaswon, M. A. and Symm, G. T. 1977. *Integral Equation Methods in Potential Theory and Elastostatics*. Academic, New York.

Kang, Y.-M., Choi, J.-H., Cho, H.-G., Lee, D.-H., and Park, C.-J. 2000. Real-time animation technique for flexible and thin objects. In *Proceedings of WSCG*, 322–329.

Kassim, A. M. A. and Topping, B. H. V. 1987. Static reanalysis: A review. *J. Struct. Eng. 113*, 1029–1045.

Kellogg, O. D. 1929. *Foundations of Potential Theory*. Springer, Berlin.

Khodakovsky, A., Schröder, P., and Sweldens, W. 2000. Progressive geometry compression. In *SIGGRAPH 2000 Conference Proceedings*, K. Akeley, Ed., Annual Conference Series. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, Reading, Mass., 271–278.

Kolarov, K. and Lynch, W. 1997. Compression of functions defined on the surface of 3D objects. In *Proceedings of Data Compression Conference*, J. Storer and M. Cohn, Eds., IEEE Computer Society Press, Los Almitos, Calif., 281–291.

Krishnamurthy, V. and Levoy, M. 1996. Fitting smooth surfaces to dense polygon meshes. *Comput. Graph. 30*, Annual Conference Series, 313–324.

Krysl, P., Lall, S., and Marsden, J. E. 2001. Dimensional model reduction in non-linear finite element dynamics of solids and structures. *Int. J. Numer. Meth. Eng. 51*, 479–504.

Kühnapfel, U., Çakmak, H., and Maass, H. 1999. 3D modeling for endoscopic surgery. In *Proceedings of IEEE Symposium on Simulation* (Delft University, Delft, NL), 22–32.

LANG, J. 2001. Deformable model acquisition and verification. PhD Thesis, Department of Computer Science, University of British Columbia.

LEE, A., MORETON, H., AND HOPPE, H. 2000. Displaced subdivision surfaces. In *SIGGRAPH 2000 Conference Proceedings*, 85–94.

LEE, A., SWELDENS, W., SCHRÖDER, P., COWSAR, L., AND DOBKIN, D. 1998. MAPS: Multiresolution adaptive parameterization of surfaces. In *SIGGRAPH 98 Conference Proceedings*, 95–104.

LOOP, C. 1987. Smooth subdivision surfaces based on triangles. MS Thesis, University of Utah, Department of Mathematics.

LOUNSBERY, M., DEROSE, T. D., AND WARREN, J. 1997. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Trans. Graph. 16*, 1 (Jan.), 34–73.

MAN, K. W., ALIABADI, M. H., AND ROOKE, D. P. 1993. Analysis of contact friction using the boundary element method. In *Computational Methods in Contact Mechanics*, M. H. Aliabadi and C. A. Brebbia, Eds. Computational Mechanics and Elsevier Applied Science, New York, Chapter 1, 1–60.

MORGENBESSER, H. B. AND SRINIVASAN, M. A. 1996. Force shading for haptic shape perception. In *Proceedings of the ASME Dynamics Systems and Control Division*, vol. 58.

PAI, D. K., VAN DEN DOEL, K., JAMES, D. L., LANG, J., LLOYD, J. E., RICHMOND, J. L., AND YAU, S. H. 2001. Scanning physical interaction behavior of 3D objects. In *SIGGRAPH 2001 Conference Proceedings*, ACM SIGGRAPH, New York.

PARAFORM. Available at http://www.paraform.com.

PENTLAND, A. AND WILLIAMS, J. 1989. Good vibrations: Modal dynamics for graphics and animation. *Computer Graphics (Proceedings of SIGGRAPH 89) 23*, 3, (Boston, July), 215–222.

PICINBONO, G., DELINGETTE, H., AND AYACHE, N. 2001. Non-linear and anisotropic elastic soft tissue models for medical simulation. In *Proceedings of IEEE International Conference on Robotics and Automation* (Seoul, Korea).

PROSKUROWSKI, W. AND WIDLUND, O. 1980. A finite element-capacitance matrix method for the Neumann problem for Laplace's equation. *SIAM J. Sci. Stat. Comput. 1*, 4 (Dec.), 410–425.

RAINDROP GEOMAGIC, INC. Available at http://www.geomagic.com.

REACHIN. Available at http://www.reachin.se.

SAID, A. AND PEARLMAN, W. A. 1996. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Trans. Circ. Syst. Video Technol. 6*, 243–250.

SCHRÖDER, P. AND SWELDENS, W. 1995a. Spherical wavelets: Efficiently representing functions on the sphere. In *Computer Graphics (Proceedings of SIGGRAPH 95)*. ACM SIGGRAPH, New York, 161–172.

SCHRÖDER, P. AND SWELDENS, W. 1995b. Spherical wavelets: Texture processing. In *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, P. M. Hanrahan and W. Purgathofer, Eds., Springer-Verlag, New York, 252–263.

SHAPIRO, J. M. 1993. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Trans. Acoustics, Speech Signal Process. 41*, 12, 3445–3462.

STAM, J. 1997. Stochastic dynamics: Simulating the effects of turbulence on flexible structures. *Comput. Graph. Forum (EUROGRAPHICS 2001 Proceedings) 16*, 3.

SWELDENS, W. 1998. The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal. 29*, 2 (Mar.), 511–546.

SZÉKELY, G., BRECHBÜHLER, C., DUAL, J., ENZLER, R., HUG, J., HUTTER, R., IRONMONGER, N., KAUER, M., MEIER, V., NIEDERER, P., ROMBERG, A., SCHMID, P., SCHWEITZER, G., THALER, M., VUSKOVIC, V., TRÖSTER, G., HALLER, U., AND BAJKA, M. 2000. Virtual reality-based simulation of endoscopic surgery. *Presence 9*, 3 (June), 310–333.

TERZOPOULOS, D. AND FLEISCHER, K. 1988. Deformable models. *Vis. Comput. 4*, 306–331.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. In *Comput. Graph. (Proceedings of SIGGRAPH 87)*, M. C. Stone, Ed, ACM, New York, 205–214.

VAN DEN DOEL, K. AND PAI, D. K. 1998. The sounds of physical shapes. *Presence 7*, 4, 382–395.

WEIMER, H. AND WARREN, J. 1998. Subdivision schemes for thin plate splines. *Comput. Graph. Forum 17*, 3, 303–314.

WEIMER, H. AND WARREN, J. 1999. Subdivision schemes for fluid flow. In *SIGGRAPH 1999 Conference Proceedings*, A. Rockwood, Ed, Addison Wesley Longman, Los Angeles, 111–120.

WU, X., DOWNES, M., GOKTEKIN, T., AND TENDICK, F. 2001. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. In *Eurographics 2001*, A. Chalmers and T.-M. Rhyne, Eds.

XIA, J. C., EL-SANA, J., AND VARSHNEY, A. 1997. Adaptive real-time level-of-detail-based rendering for polygonal models. *IEEE Trans. Vis. Comput. Graph. 3*, 2, 171–183.

YOSHIDA, K., NISHIMURA, N., AND KOBAYASHI, S. 2001. Application of fast multipole Galerkin boundary integral equation method to elastostatic crack problems in 3D. *Int. J. Numer. Meth. Eng. 50*, 525–547.

YSERENTANT, H. 1986. On the multilevel splitting of finite element spaces. *Numer. Math. 49*, 379–412.

ZHUANG, Y. AND CANNY, J. 2000. Haptic interaction with global deformations. In *Proceedings of the IEEE International Conference on Robotics and Automation* (San Francisco).

ZIENKIEWICZ, O. C. 1977. *The Finite Element Method*. McGraw-Hill, Maidenhead, Berkshire, England.

ZILLES, C. B. AND SALISBURY, J. K. 1994. A constraint-based god-object method for haptic display. In *ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems*, vol. 1 (Chicago), 149–150.

ZORIN, D. AND SCHRÖDER, P., Eds. 2000. *Course notes: Subdivision for modeling and animation*. ACM SIGGRAPH, New York.

Received December 2001; revised July 2002; accepted August 2002