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

Adaptive Deformations with Fast Tight Bounds

Miguel A. Otaduy[1], Daniel Germann[1], Stephane Redon[2] and Markus Gross[1]

[1]Computer Graphics Laboratory, ETH Zurich, Switzerland
[2]i3D - INRIA Rhone-Alpes, Grenoble, France

Abstract

Simulation of deformations and collision detection are two highly intertwined problems that are often treated separately. This is especially true in existing elegant adaptive simulation techniques, where standard collision detection algorithms cannot leverage the adaptively selected degrees of freedom. We propose a seamless integration of multigrid algorithms and collision detection that identifies boundary conditions while inherently exploiting adaptivity. We realize this integration through multiscale bounding hierarchies*, a novel unified hierarchical representation, together with an adaptive multigrid algorithm for irregular meshes and an adaptivity-aware hierarchical collision detection algorithm. Our solution produces detailed deformations with adapted computational cost, but it also enables robust interactive simulation of self-colliding deformable objects with high-resolution surfaces.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Physic. based modeling

1. Introduction

Simulation of deformable bodies has long attracted the attention of computer graphics, and today we can find sophisticated methods for highly-realistic simulation of e.g., human biomechanics [ITF04], cloth [BFA02], or fracture effects [OBH02] among others. In later years, deformable bodies have also seen application in videogames or virtual surgery, through the development of plausible interactive simulation techniques [JP03, BJ05, MHTG05].

Contact exacerbates the complexity of simulating deformable bodies, by exciting arbitrarily many deformation modes and inducing rich and diverse deformations. To attack the complexity of the problem, adaptive simulation methods [DDCB01, GKS02, CGC*02] employ fewer degrees of freedom (DoFs) on regions that allow for lower resolution. However, in computer graphics, the boundary of the simulation domain (i.e., the surface) plays a central role, and handling of boundary conditions for producing accurate surface deformations requires the use of high-resolution surfaces. Collision detection becomes then a central component of the simulation, as part of the process for setting boundary conditions. To the best of our knowledge, there is no prior technique that bridges the simulation of deformations and collision detection, and that exploits adaptivity for setting (i.e., detecting) boundary conditions. The BD-Tree [JP04] and reduced deformable models have been combined to produce an elegant solution for cases where deformations are described by a few constant DoFs, but can hardly handle some of the rich effects produced by contact deformation dynamics.

Figure 1: ***Interactive Deformation with Self-Collisions.*** *A dragon with* 13480 *triangles deforms interactively (almost* 10 *fps on average), with accurate handling of self-collisions.*

In this paper, we propose a seamless integration of multigrid methods [Bra77, BHM00] and hierarchical collision detection [GLM96] for exploiting adaptivity when setting boundary conditions in the context of contact and deformation simulations. This integration relies on three main contributions:

- **Multiscale bounding hierarchy (MBH):** A unified representation that combines a hierarchy of irregular non-nested tetrahedral meshes and a bounding volume hierarchy (BVH), and maps tetrahedra at multiple scales to the BVs.

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

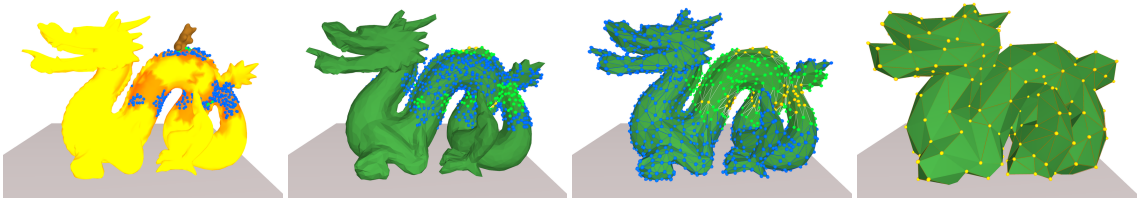


Figure 2: ***Adaptive Refinement of a Mesh Hierarchy.*** *Left: surface of the fine mesh M_1 of the dragon model, color-coded by the local active resolution ('yellow' is coarse, 'red' is fine). Rest: boundary of tetrahedral meshes M_2, M_3, and M_5 with interior (orange), front (green), and interface nodes (blue). Notice the change of genus in the hierarchy.*

• **Adaptive multigrid on irregular meshes:** Through careful augmentation of mesh data structures and refinement/coarsening strategies, the algorithm is as simple as adaptive multigrid for regular nested meshes [Bra77], but conforms well to rich object boundaries using non-nested irregular meshes. We have applied it to corotational linear elasticity with implicit integration [MG04].

• **Adaptivity-aware (self-)collision detection:** Exploiting the MBH to perform efficient on-demand refitting of bounding volumes, the cost for collision (and self-collision) detection depends on the active DoFs and the number of (potential) contacts, not surface complexity.

We demonstrate our approach on collision-intensive multi-object scenes (See Fig. 4), and accurate computation of contact manifolds in self-collions (See Fig. 5). But, most importantly, and as shown in Fig. 1, the seamless connection between simulation of deformations and collision detection allows for efficient and robust handling of (self-)collisions with high-resolution surfaces at interactive rates.

In §2 we discuss related work, and in §3 we formulate the dynamic deformation problem at hand, and we discuss its solution with adaptive multigrid on regular nested meshes. In §4 we describe our adaptive solution for irregular meshes, based on an elegant augmentation of the data structures and refinement/coarsening strategies that permits the use of the very same algorithm as for regular nested meshes. In §5 we describe the MBH in detail, and we present our collision detection algorithm, exploiting adaptively selected DoFs. We also present an extremely fast method for computing tight bounds for sets of points interpolated inside a simplex. We finally discuss the results of our experiments, as well as possible directions of future work.

2. Related Work

We focus here on methods for adaptive simulation of deformable objects and collision detection. As our paper introduces no novel deformation model, we refer the reader to comprehensive surveys for details [GM97, NMK*05].

Grinspun et al. [GKS02] developed a general method for adaptive simulation with the finite element method (FEM) in computer graphics, with the advantage that adaptivity is achieved through refinement of basis functions, not decomposition of mesh elements. Their method requires subdivision connectivity of the meshes. Capell et al. [CGC*02] proposed another technique built on subdivision, but they embedded the surface of the object to be deformed. Their technique allows, for example, the application of constraints on the embedded mesh. Debunne et al. [DDCB01], instead, designed a method for adaptive simulation of deformations using non-nested irregular meshes, which conform more accurately to the boundary with fewer elements. A fundamental difference of our multigrid approach is a hierarchical definition of the deformation problem, which naturally enables a link between the data structures for adaptive simulation and collision detection. Moreover, in Debunne's approach, interpolation at the interface between different mesh resolutions is not symmetric, therefore making the method not well suited for full implicit simulation with fast iterative solvers. Similar to adaptive methods, reduced models provide very fast simulations with a few DoFs that drive complex surfaces [BJ05]. However, their strength is in global deformations, not accurate surface deformations during contact.

Multigrid algorithms were initially intended (and show optimal linear convergence) for solving elliptic boundary value PDE problems. Nonetheless, they see application on a much larger range of problems, and under multiple variants. We refer to books on the topic [BHM00, TOS01] for a comprehensive treatment. The common feature of multigrid algorithms is to solve the problem by smoothing the error through iterative relaxation on different scales. The typical implementation of multigrid is based on a correction scheme that filters the fine-scale residual and corrects it on a coarser scale. Brandt [Bra77] introduced the full approximation storage (FAS) scheme, which filters both the residual and the fine-scale estimate of the solution, thus correcting the full solution on the coarser scale. FAS allowed Brandt to implement multilevel adaptive technique (MLAT) solutions on nested regular meshes. The fast adaptive composite (FAC) grid method [MT86] constitutes another popular adaptive multigrid technique, but it relies strongly on the use of regular meshes. Little work exists in the application of adaptive multigrid to irregular meshes in 3D, due to the difficulty in maintaining interface regions of locally refined meshes, and smoothly interpolating values across meshes.

In the context of computer graphics, multigrid has been used for simulating thin shells [GTS02], enhancing the stability

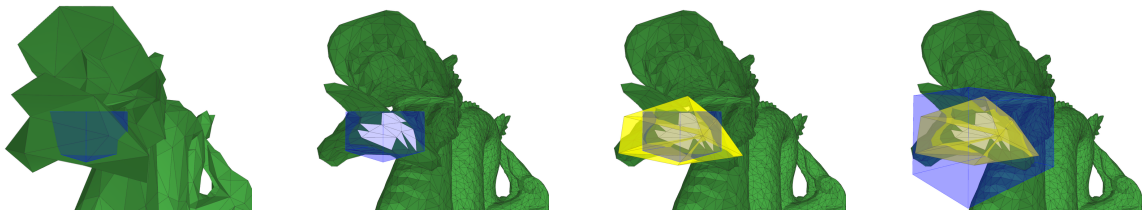


Figure 3: ***Multiscale Bounding Hierarchy (MBH).*** *From left to right: (i) Mesh M_4 of the dragon model, and one bounding volume B (in blue) at the corresponding level L_4 of the BVH. (ii) Finest mesh M_1, and the surface patch (in white) bounded by B. (iii) When M_4 is the locally finest active mesh, B can be updated directly from its effective governor tetrahedra $G^*(B) \in M_4$ (superimposed in yellow), without accessing or evaluating the finest surface ∂M_1. (iv) Notice the loose AABB computed from the tetrahedra, as opposed to the tight AABB computed with our method based on simplex interpolation (in the other images).*

of explicit integrators [WT04], accelerating the simulation of deformable models in a non-adaptive manner [GW05], or recently for fast mesh deformations in geometric modeling [SYBF06]. Aksoylu et al. [AKS05] have designed general multigrid solvers on unstructured meshes.

Collision detection of deformable objects has been addressed using techniques such as: BVHs [dB97], spatial hashing [THM*03], visibility-based culling [GRLM03], or second-order Voronoi diagrams [SGG*06] (See [TKH*05] for a recent survey). Self-collision further complicates the problem [VMT94, VMT06], as primitive adjacency is difficult to distinguish from actual collisions. For a surface with n primitives, most techniques have a best-case $O(n)$ cost. The BD-tree [JP04] for linear parametric models constitutes a notable exception. It may prune non-colliding objects by testing only the root of the tree, which can be updated with a cost linear in the number of deformation modes, not the size of the surface. Our multiscale bounding hierarchy (MBH) is also optimal in the sense that its best-case update cost (i.e., for a non-colliding object) is linear in the number of DoFs of the simulation. As opposed to hybrid approaches for updating BVHs [LAM01], it presents a $O(1)$ cost for tightly updating bounding volumes on demand. Moreover, by combining features of BVHs and spatial hashing, it also allows for fast hierarchical pruning in self-collision detection. Last, visual or haptic perceptual metrics [ODGK03, OL03] can be incorporated for implementing interruptible or adaptive collision detection.

3. Adaptive Multigrid Deformations

In this section we describe the discretized dynamic deformation problem, its multigrid solution with the FAS scheme, and the addition of adaptivity with the MLAT algorithm for regular nested meshes, This algorithm sets the foundations for our extension to irregular non-nested meshes, which we will describe in §4.

3.1. Discretized Deformation Dynamics

Linear FEM discretization of linear-elastic deformation dynamics leads to motion equations $\mathbf{M}\ddot{\mathbf{x}} + \mathbf{D}\dot{\mathbf{x}} + \mathbf{K}\mathbf{x} = \mathbf{F}_{ext}$, where $\mathbf{M}$, $\mathbf{D}$, and $\mathbf{K}$ are, respectively, mass, damping, and stiffness matrices, and $\mathbf{F}_{ext}$ is a vector of external forces (See [NMK*05] for more details). We apply lumping to yield diagonal $\mathbf{M}$ and $\mathbf{D}$ matrices, and per-tetrahedron stiffness warping [MG04] to improve the behavior of linear elasticity under large deformations. Backward Euler implicit integration with a first order approximation of forces [BW98] yields the following linear system for computing node positions:

$$\mathbf{A}\mathbf{x}_i = \mathbf{b}, \tag{1}$$

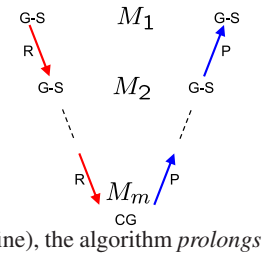
$$\mathbf{A} = \mathbf{M} + \Delta t\mathbf{D} + \Delta t^2\mathbf{K}, \tag{2}$$

$$\mathbf{b} = \mathbf{A}\mathbf{x}_{i-1} + \Delta t\,(\mathbf{M} + \Delta t\mathbf{D})\,\mathbf{v}_{i-1} + \Delta t^2\mathbf{F}_{i-1}. \tag{3}$$

3.2. FAS Multigrid

The first component of our novel multiscale bounding hierarchy (MBH) is a sequence of non-nested irregular tetrahedral meshes $\mathscr{M} = \{M_1, M_2, \ldots M_m\}$ (See an example in Figure 2). Unlike the common numbering convention in the literature, we denote by M_1 the highest resolution mesh, to establish a clear correspondence between levels of $\mathscr{M}$ and the BVH. In this section, and w.l.o.g., we will consider $\mathscr{M}$ as a sequence of nested regular meshes. We assume that the deformation field animates a high-resolution surface S, and, for simplicity, throughout the paper we assume $S = \partial M_1$.

We have implemented a V-cycle of FAS multigrid [BHM00, TOS01] on the hierarchy $\mathscr{M}$ of meshes to solve the discretized linear problem (1). FAS computes an approximation to the full position $\mathbf{x}$ on every mesh (i.e., approximations with different frequency components), and this property is important for computing bounds of $\mathbf{x}$ based on coarse mesh information, and thus designing the BVH. On the top-down pass (from fine to coarse), the algorithm successively relaxes the error on each level using Gauss-Seidel (G-S) iteration, and *restricts* (R) the residual $\mathbf{b} - \mathbf{A}\mathbf{x}$ to the next coarser level. On the bottom-up pass (from coarse to fine), the algorithm *prolongs* (P) the coarse corrected solution to the next finer level and performs further error relaxation. On the coarsest level, we use a Conjugate Gradient (CG) iterative solver.



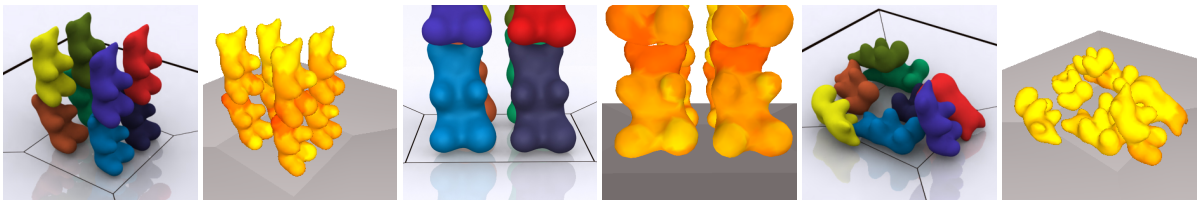


Figure 4: ***Adaptively Deformed Gummy Bears.*** *Each bear has a 6-mesh hierarchy (from 85 to 6418 tetrahedra). Notice the refinement (red) when collisions induce high stress, and the coarsening (yellow) when the bears come to rest. Our adaptive algorithm offers about 10x speed-up over full-resolution conjugate gradient or non-adaptive multigrid. See details in Table 1.*

In FAS, the right-hand side on a coarse mesh M_{j+1} is formulated using a *residual restriction operator* $\mathbf{R}$, and a *solution restriction operator* $\tilde{\mathbf{R}}$ as:

$$\mathbf{b}^{j+1} = \mathbf{R}^{j+1}\left(\mathbf{b}^j - \mathbf{A}^j\mathbf{x}^j\right) + \mathbf{A}^{j+1}\tilde{\mathbf{R}}^{j+1}\mathbf{x}^j. \quad (4)$$

The corrected solution is prolonged to M_j by a *prolongation operator* $\mathbf{P}$ as:

$$\mathbf{x}^j \leftarrow \mathbf{x}^j + \mathbf{P}^j(\mathbf{x}^{j+1} - \tilde{\mathbf{R}}^{j+1}\mathbf{x}^j). \quad (5)$$

On the finest mesh, we use as initial estimate

$$\mathbf{x}_i^1 \leftarrow \mathbf{x}_{i-1}^1 + \Delta t\,\mathbf{v}_{i-1}^1. \quad (6)$$

On coarser meshes M_{j+1}, we use as initial estimate the restricted solution, $\tilde{\mathbf{x}}^{j+1} = \tilde{\mathbf{R}}^{j+1}\mathbf{x}^j$.

We define $\mathbf{A}^j$ by discretizing the dynamic deformation problem on every mesh, which provides a formulation independent of the level of refinement in the adaptive setting. We leave the description of the interpolation operators $\mathbf{P}$, $\mathbf{R}$, and $\tilde{\mathbf{R}}$ to the specific case of irregular tetrahedral meshes in §4.1.

3.3. MLAT Algorithm

In MLAT, each mesh M_j is active on a domain Ω_j, and may be refined locally to yield an active domain $\Omega_{j-1} \subseteq \Omega_j$ on the finer mesh M_{j-1}. We define as active nodes those that belong to an active domain Ω. Error relaxation should be executed on the active domains Ω_j, which requires the evaluation of values on inactive nodes on the *interface* $\partial\Omega_j$. We further divide active and inactive nodes into four subclasses:

- *Interior*: active nodes on regions that are refined.
- *Front*: active nodes on regions that are not further refined.
- *Interface*: inactive nodes where the function $\mathbf{x}$ needs to be evaluated in the context of the multigrid algorithm.
- *Idle*: all other (inactive) nodes.

Values on interface nodes should be interpolated from the coarser active regions in which they lie. On regular nested meshes, coarse nodes are simply a subset of fine nodes, and interpolation of interface nodes is straightforward [Bra77, TOS01], because the nodes on a coarse mesh M_{j+1} that define the value $\mathbf{x}$ on an interface node of the next finer mesh M_j are by construction active or interface themselves. The FAS algorithm can be modified to account for active subdomains and adaptivity, leading to the MLAT algorithm:

Algorithm 1 ***V-Cycle of MLAT.***

```
//Top-Down Pass
  for meshes M_j, j = 1 → m − 1,
    Initialize x^j on front nodes based on (6).
    Set b^j on front nodes based on (3).
    Perform G-S pre-relaxation on x^j on active nodes.
    Restrict x̃^{j+1} ← R̃^{j+1} x^j to interior nodes.
    Restrict b^{j+1} to interior nodes as in (4).
//Coarse Mesh Solution
  Solve A^m x^m = b^m with Conjugate Gradient.
//Bottom-Up Pass
  for meshes M_j, j = m − 1 → 1,
    Interpolate x^j on interface nodes based on P.
    Correct x^j by prolongation (5) on active nodes.
    Perform G-S post-relaxation on x^j on active nodes.
```

4. MLAT on Irregular Meshes

In this section we describe our extension of MLAT to irregular meshes, through appropriate interpolation operators, an augmentation of node neighborhoods, and simple refinement and coarsening strategies. We also describe our error criterion, the interpolation of the deformation to the animated surface, and constraint-based collision response.

4.1. Irregular Meshes and Interpolation Operators

We construct the meshes $\{M_j\}$ in a way such that their boundaries approximate the animated surface S, and all vertices of S are contained in every mesh M_j in the rest configuration. Note, however, that we do not enforce topological equivalence across meshes (e.g., the meshes may have different genus) (See Figure 2 for an example). To construct each mesh M_j, we start from the triangle mesh S, create an offset surface, apply triangle mesh decimation, and mesh the interior with an off-the-shelf tetrahedral mesh generator.

Prolongation $\mathbf{P}$ can be regarded as upsampling, and is typically implemented as interpolation. Specifically, for every node $N \in M_j$, we identify the tetrahedron $T \in M_{j+1}$ that is closest to or encloses N. W.l.o.g., we refer to it as the *coarse enclosing* tetrahedron T^+ of N, $T = T^+(N)$. We define the corresponding row of $\mathbf{P}^j$ by the barycentric coordinates of N in $T^+(N)$ in the rest configuration. For the residual restriction operator $\mathbf{R}$, we follow the common choice of

$\mathbf{R}^{j+1} = \left(\mathbf{P}^j\right)^T$ [BHM00]. For the solution restriction operator $\tilde{\mathbf{R}}$, however, we choose barycentric interpolation of each coarse node $N \in M_j$ in the closest or enclosing tetrahedron $T \in M_{j-1}$. W.l.o.g., we refer to it as the *fine enclosing* tetrahedron T^- of N, $T = T^-(N)$.

On irregular meshes, we identify the following sources of potential inconsistencies in the cross-mesh interpolations of Algorithm 1, which we avoid through the augmentation of mesh data structures and careful refinement and coarsening strategies presented in the next subsection:

- Prolongation to an active node $N \in M_j$ according to (5) in the bottom-up pass of Algorithm 1 may use inactive nodes in M_{j+1} in a straightforward application to irregular meshes. In order to satisfy appropriate convergence of the multigrid algorithm, the algorithm must use only active nodes in M_{j+1}. Moreover, the correction $\mathbf{x} - \tilde{\mathbf{x}}$ is conceptually a correction to the residual only for interior coarse nodes in M_{j+1}. For front coarse nodes, we consider $\tilde{\mathbf{x}}$ simply as the initial value in the top-down pass of the V-cycle.
- Interpolation to an interface node may use idle nodes without a proper solution estimate, and correct interpolation must use only active or interface nodes.
- Similarly, restriction to an interior node may use idle nodes, but must use only active or interface nodes. In the residual restriction to a node $N \in M_j$ according to (4), we set a value 0 for the residual of interface nodes in M_{j-1}.

4.2. Refinement and Coarsening

To guide the refinement and coarsening, we construct a *node forest* $\mathcal{N}$ similar to the one of Debunne et al. [DDCB01]. For each node $N \in M_j$, we set as its parent in $\mathcal{N}$ the closest node from $T^+(N) \in M_{j+1}$. Note that Debunne's forest definition would not avoid the interpolation inconsistencies.

Furthermore, we define a set of *cascade neighbors* $C(N)$ on each node N, which conceptually augments the 1-ring neighborhood of N, and can be precomputed without additional run-time bookkeeping. Specifically, N_j belongs to $C(N)$ if at least one of the following conditions holds: (a) N_j belongs to the 1-ring neighborhood of N; (b) given a 1-ring neighbor N_k of a child of N, N_j is one of the nodes in $T^+(N_k)$; (c) N_j is the parent in $\mathcal{N}$ of a node in $T^-(N)$; and/or (d) N is a cascade neighbor of N_j (to ensure symmetry of the definition). We can now introduce the following simple refinement and coarsening strategies for irregular meshes:

Refinement: If a front node N does not satisfy the error criterion (See §4.3), we refine it by converting it into an interior node, and activating all its children in $\mathcal{N}$. At the same time, we enforce that all nodes in $C(N)$ must be active, which may induce cascading refinements.

Coarsening: If all children of an interior node N are front, none of their cascade neighbors is an interior node, and N satisfies the error criterion, we coarsen N by converting it into a front node, and deactivating all its children in $\mathcal{N}$.

Our definition of node forest, cascade neighbors, and simple refinement and coarsening strategies enable the use of the very same MLAT Algorithm 1 on irregular meshes. Refinement is executed after a complete execution of Algorithm 1, and starts on the coarsest mesh M_m. Coarsening is executed after refinement, and starts on the finest mesh M_1. After refinement and coarsening are completed, we initialize the interface for the next time step, by setting inactive neighbors and children of front nodes as interface, and initializing $\mathbf{x}$ on them in a bottom-up pass (from coarse to fine).

4.3. Error Criterion

We determine the locally required resolution of the multigrid hierarchy $\mathcal{M}$ by comparing the solution on the two finest active levels. Specifically, and as common in adaptive finite element simulations [Ran99], we compute on a front node $N \in M_j$ a local residual that weights by the local linear system $\mathbf{A}$ the difference between the local solution and the one prolonged from the next coarser level:

$$e(N) = \|\mathbf{A}^j(\mathbf{x}^j - \mathbf{P}^j\mathbf{x}^{j+1})\|. \tag{7}$$

Our error criterion measures the local quality of the linear approximation on M_{j+1}, and maintains M_j as a conservative level for error computations. By weighting the solution difference by $\mathbf{A}$, the error criterion depends on the local stiffness of the equations, thus reducing possible popping due to dynamic refinement and coarsening. We compare $e(N)$ to predefined error thresholds, but they could also be weighted by distance to the camera, amount of occlusion, or size of the contact area, to account for visual or haptic perception [ODGK03, OL03]. Note that the error threshold may not be too high, or popping will be inevitable.

4.4. Surface Evaluation and Collision Response

Application of boundary conditions in the dynamic simulation requires the definition of positions $\mathbf{x}$ on the animated surface S. Note that, for collision detection, we compute bounds of regions of S, and $\mathbf{x}$ is explicitly evaluated only on colliding points. We define a tetrahedron T as *front* if at least one of its nodes is a front node. Then, we define the position $\mathbf{x}$ of a vertex $V \in S$ as a barycentric combination of the nodes of its finest enclosing front tetrahedron (in rest configuration), denoted by $T^F(V)$. Due to the containment of S on all multigrid meshes, $\mathbf{x}$ is strictly defined as a *convex* combination of nodes, and this is crucial for designing the BVH (See §5).

Our adaptive multigrid solution to dynamic deformations allows the use of diverse contact handling methods, such as penalty-based or constraint-based. In our simulations, we have followed the approach of solving a linear complementarity problem (LCP) using the projected Gauss-Seidel method [CPS92]. We first solve a collision-free version of the discretized forward dynamics problem (1), then we execute continuous collision detection [RKC02] to detect and formulate (inequality) contact constraints, and finally solve

Figure 5: ***Self-Collisions in a Knot.*** *A knot with* 42 254 *triangles and a* 7*-mesh hierarchy (*119 399 *tets in* M_1*) is dropped and suffers large self collisions. The middle image shows the resolution near the surface ('yellow' is coarse, 'red' is fine), achieving smooth deformations and accurate handling of contact boundary conditions. Table 2 evaluates collision detection performance.*

the LCP. Figure 5 shows smooth and robust contact surfaces that can be achieved by combining an LCP formulation with our simulation and collision detection approach. Constraint-based collision response is costly when applied to high-resolution objects undergoing a large number of contacts, but our multigrid hierarchy accelerates response computations, as each contact force is applied directly on the nodes of $T^F(V)$ for each colliding vertex V, and the effect propagates quickly through the object on the coarse meshes.

5. Adaptivity-Aware Collision Detection

We now complete the definition of the MBH, as we describe the multiscale connections between BVs and tetrahedra, the fast fitting of tight bounds for sets of points interpolated in a simplex, and our novel adaptivity-aware self-collision detection. We first introduce the concepts for a non-adaptive simulation with a high-resolution surface S embedded in a coarse simulation mesh M_f, and later extend them to the adaptive setting. Our algorithms are independent of the type of BV, but here we discuss an AABB-based implementation.

5.1. Multiscale Bounding Volumes

Our novel MBH consists of the sequence of non-nested irregular tetrahedral meshes $\mathscr{M} = \{M_1, M_2, \dots M_m\}$ introduced in §3.2, and the levels of a BVH $\mathscr{B} = \{L_1, L_2, \dots L_l\}$ (with M_1 and L_1 the finest mesh and level). For bounding the motion of the animated surface S, we enforce that all meshes M_j and BVH levels L_j bound S.

The BVH $\mathscr{B}$ is a tree of *multiscale bounding volumes* (MBVs) that bound S in a hierarchical manner. Conceptually, we set a bidirectional correspondence between the lower m levels of $\mathscr{B}$ and the meshes in $\mathscr{M}$. Classical BVH refitting first computes the leaf BVs and then updates the rest of the hierarchy in a bottom-up manner. The intuitive idea behind our adaptivity-aware collision detection algorithm is to exploit the correspondences between levels of $\mathscr{B}$ and meshes of $\mathscr{M}$, to refit coarse MBVs directly and in a fast manner (i.e., with cost $O(1)$) using active tetrahedra.

We define the tree connectivity of $\mathscr{B}$ as a preprocessing operation, by successive splitting of the triangles of S at the median of the longest axis defined by the covariance matrix [GLM96]. Let us assume, w.l.o.g., that this procedure yields a perfectly binary tree with l levels. In this way, a MBV B at level L_j must bound a set of 2^{j-1} triangles in S.

5.1.1. Governors for Efficient Dynamic Refitting

Let us start with the assumption that the front of $\mathscr{M}$ is fixed at a coarse mesh M_f (i.e., only M_f and coarser levels are active) and later relax this assumption in §5.4. Given M_f, we similarly define its corresponding BVH level L_f as the front of $\mathscr{B}$. For each MBV $B \in L_j, j \leq f$, we define the *effective governor* tetrahedra $G^*(B) \in M_f$ as those that bound (in the rest configuration) the set of vertices $V(B) \in S$ that must be bounded by B. Figure 3 shows one BV $B \in L_4$, its governors $G^*(B) \in M_4$, and the patch it bounds in $S = \partial M_1$. When the locally finest active mesh is $M_f = M_4$, we can detect collisions using only DoFs from M_4, without ever evaluating M_1.

5.1.2. Refitting Algorithm

We exploit the knowledge about the front of $\mathscr{M}$ to refit $\mathscr{B}$ with an adaptivity-aware cost in the following manner. Before a collision detection query, we bound MBVs in L_f using effective governors. Then, we bound all other MBVs in the subtree $\mathscr{B}_f = \{L_j, j \geq f\}$ in a bottom-up manner.

During a collision detection query, we compute bounds on demand for each visited MBV B below the front L_f, using its effective governors $G^*(B) \in M_f$. In simulations with temporal coherence, we cache the front of the subtree $\mathscr{B}^*$ of $\mathscr{B}$ visited during the previous collision query. Then, before the next query, we apply the adaptivity-aware refitting to the front of the subtree $\mathscr{B}^* \cup \mathscr{B}_f$ instead.

Our refitting algorithm is optimal if the contribution of each governor to the bounds can be computed in $O(1)$ time and every MBV $B \in L_f$ has $O(1)$ governors. In the next subsection we present an efficient $O(1)$ computation of tight bounds. The number of governors can be controlled during the meshing process, by ensuring low enough sampling at each mesh M_j. As a reference, the required mesh size ratio is $\frac{\|M_j\|}{\|M_{j+1}\|} \geq \sqrt{8}$ for a cube with uniform sampling.

5.2. Tight Bounds in Simplex Interpolation

We now present a novel and fast method for computing tight bounds along a direction x of a set of points V interpolated inside a simplex T (in our case, a tetrahedron). It is more efficient than evaluating the points even for very small sets, and often yields much tighter bounds than simply bounding the simplex. We use it for computing tight AABBs in the dynamic refitting of MBVs on or below the front of $\mathscr{B}$.

Using planes of minimum and maximum barycentric coordinates in rest configuration, we define a convex polyhedron that bounds V and is entirely contained in T. This polyhedron constitutes an 8-DOP for a tetrahedron, and the same concept has been used before for bounding skinned surfaces [KZ05]. Given vertices $\{V_A, V_B, V_C, V_D\}$, the 8-DOP is defined by barycentric intervals $\{[a_A, b_A], [a_B, b_B], [a_C, b_C], [a_D, b_D]\}$ (See Figure 6 (left) for a 2D example). At runtime computation of the AABB, we conservatively bound V by bounding the 8-DOP instead.

Instead of evaluating the corners of the 8-DOP, we propose a novel method that sorts the vertices of T and computes bounds in close-form. Note that every front tetrahedron constitutes typically an effective governor for many MBVs, but it only needs to be sorted once per time step. For a direction x, we sort the vertices of T, resulting in $x_0 \leq x_1 \leq x_2 \leq x_3$ for a tetrahedron (Figure 6 (right) shows a 2D example). Then, we efficiently compute AABB bounds $[a_x, b_x]$ by indexing the barycentric intervals according to the sorted vertices as:

$$a_x = b_0 x_0 + a_3 x_3 + a_2 x_2 + (1 - b_0 - a_3 - a_2) x_1,$$
$$b_x = b_3 x_3 + a_0 x_0 + a_1 x_1 + (1 - b_3 - a_0 - a_1) x_2. \quad (8)$$

This procedure is not guaranteed to optimally bound the 8-DOP, as it may make a wrong, but conservative, assumption on the intersection of barycentric planes that define the 8-DOP corners. However, in practice it provides very tight bounds, as demonstrated in Figure 3. The method is extensible to simplex interpolation in higher (or lower) dimensions. It is important to note that the 8-DOP may not be replaced by an AABB or an OBB in rest configuration, because their corners may not be defined by convex combination in T, therefore they may not bound V during deformation.

5.3. Self-Collision Handling

We also exploit knowledge about the multigrid hierarchy $\mathscr{M}$ to devise an adaptivity-aware test for self-collision detection. We combine features of spatial hashing [THM*03] and BVHs to design a self-collision detection algorithm with a best-case (i.e., no collision) cost linear in the number of front tetrahedra near the surface. If the front mesh M_f is notably coarser than the surface S, our algorithm largely outperforms those linear in the size of S. Moreover, performance degrades gracefully under actual self-collisions, and many areas are pruned with a minor cost as shown in Figure 8.

5.3.1. Potentially Self-Colliding Patches

Given a front mesh M_f that contains S and does not self-intersect in the rest configuration, let us first define a patch $S_a \subset S$ as *potentially self-colliding* if one (or more) of its effective governors $G^*(S_a) \in M_f$ intersects some other tetrahedron in M_f. Then, our algorithm builds on the following lemma: Given two patches $S_a, S_b \subset S$ that do not intersect in the rest configuration, S_a and S_b do not collide if at least one of them is not potentially self-colliding. The same lemma is valid for a self-collision (S_a, S_a). We do not consider tetrahedra that share only a vertex, an edge, or a face, as intersecting, thus pruning potential self-collisions due to adjacency.

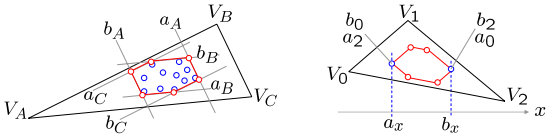


Figure 6: ***Tight Fitting k-DOP.*** *Left: 2D example showing a set of vertices (in blue) enclosed in a triangle, and the 6-DOP (in red) defined by barycentric coordinate bounds. Right: To compute the bounds $[a_x, b_x]$ of the 6-DOP along x, it suffices to sort the vertices of the triangle and evaluate a close-form definition of the extreme corner of the 6-DOP.*

5.3.2. Hierarchical Self-Collision Query

Before a self-collision query, we perform a CCD intersection test (using spatial hashing) between front tetrahedra of M_f that are effective governors of some MBV (They are the tetrahedra enclosing S). Then, we flag a front MBV $B \in L_f$ as potentially self-colliding if at least one of its effective governors $G^*(B)$ intersects some other front tetrahedron. We propagate the information up in $\mathscr{B}$ by flagging a MBV as potentially self-colliding if at least one of its children is flagged. If a collision query descends below the front L_f, we flag MBVs using information from the effective governors directly.

Hierarchical self-collision pruning can then be easily implemented as follows. A self-collision test (B_a, B_a) can be pruned if B_a is not potentially self-colliding. Otherwise, one must split B_a and test its children. Notice that, in most hierarchical algorithms, a test (B_a, B_a) can never be pruned ([VMT94] is an exception, but its cost is linear in the size of S), but in our case a self-collision query may be pruned right at the root of the BVH! A self-collision test (B_a, B_b) can be pruned if at least one MBV is not potentially self-colliding, or if the AABBs do not collide (as in a regular query).

5.4. Extension to Adaptive Meshes

With adaptive meshes, the front of $\mathscr{M}$ may combine multiple meshes M_j, therefore effective governors cannot be known a priori. We extend the basic BVH refitting algorithm described in §5.1 by including governors at multiple levels, a definition of BVH front for the adaptive setting, and an algorithm for efficiently determining effective governors.

We extend the self-collision algorithm by flagging a MBV as potentially self-colliding if some effective governor intersects other tetrahedra, or the effective governors lie on different mesh levels. The spatial-hashing test between front tetrahedra must be carried out at each level independently.

5.4.1. Multiscale Governors

As a preprocessing, and for every MBV $B \in L_j, j \leq m$, we identify the *governor* tetrahedra $G(B)$ at coarser levels,

$G(B) \in \{M_k\}, m \geq k \geq j$, that may contribute to the dynamic definition of the bounds of B, and we store them in a directed graph, as shown in Figure 7 for a 2D example. We set an edge from a governor tetrahedron $T_a \in M_j$ to a coarser governor tetrahedron $T_b \in M_{j+1}$, if both T_a and T_b bound at least one common vertex in the set $V(B)$ to be bounded by B.

5.4.2. Definition of the BVH Front

A MBV $B \in L_j$ is a *potential front* MBV if at least one of its governors at the corresponding level, $T \in G(B) \cap M_j$, is a front tetrahedron. Then, B is a front MBV if it is a potential front MBV and it has no potential front descendants. With this definition, the front of $\mathscr{B}$ is formed by roughly as-coarse-as-possible MBVs that can be conservatively refit with $O(1)$ cost each, and with an adaptivity-aware cost altogether, i.e., linear in the number of front tetrahedra in $\mathscr{M}$.

To find the front of $\mathscr{B}$ at runtime, we first mark the MBVs governed by each front tetrahedron at its corresponding level. Then we perform a bottom-up flooding toward the root of $\mathscr{B}$, marking other visited MBVs, and finally we perform a top-down pass to detect the boundary of the marked region.

5.4.3. Effective Governor Tetrahedra

In order to compute the bounds of a front MBV B, we seek to identify those governors that effectively define the position $\mathbf{x}$ of some surface vertex V bounded by B. We refer to them as *effective governors* $G^*(B)$, and they are, according to §4.4, the finest active enclosing tetrahedra $\{T^F\}$ of all vertices bounded by B. Given a front MBV $B \in L_j$, we find $G^*(B)$ by performing a bottom-up search on $G(B)$, seeded at governors from the corresponding mesh, $\{T\} \in G(B) \cap M_j$.

Finding effective governors varies slightly for MBVs updated on-demand during a collision detection query. In the traversal of $\mathscr{B}$ during the query, every MBV B passes to its children the level j of its finest effective governor. Then, the children seed the search for finding their effective governors at level j in the governor graph. The rest of the fitting algorithm is identical as for front nodes.

6. Results

We have performed all our simulations on a dual 3.2 GHz. processor PC with 2 GB of memory. They include a scene with multiple gummy bears suffering impacts and deformations (See Figure 4), a knot falling on the ground and experiencing self-collisions (See Figure 5), and the interactive deformation of a dragon model also undergoing self-collisions (See Figures 1 and 8).

The bears experience drastic yet smooth and stable changes in the refinement of the multigrid hierarchy, due to impact-induced deformations, as can be clearly seen in the accompanying video. Each bear is described by a 6-mesh hierarchy, from 85 tetrahedra in M_6 to 6418 in M_1. We have compared our adaptive algorithm to full resolution conjugate gradient and a non-adaptive FAS multigrid solver, as shown in

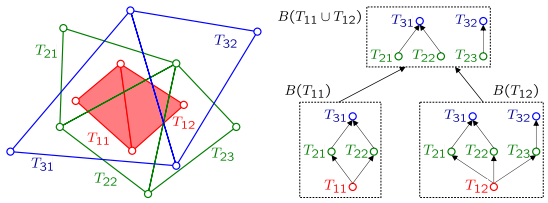


Figure 7: ***MBV Governors in 2D.*** *Left: Two triangles, $T_{11} \in M_1$ and $T_{12} \in M_1$ (in red), and their bounding triangles in M_2 (in green) and M_3 (in blue); Right: Graphs of governors for the MBVs $B(T_{11})$, $B(T_{12})$, and $B(T_{11} \cup T_{12})$.*

Table 1, in scenarios with 1 and 8 bears. The first conclusion is that FAS multigrid imposes no overhead on the solution. Second, our adaptive algorithm offers about 10x average speed-up both in the dynamics update as in the complete simulation, even though the bears reach full resolution (locally) at times. Constraint-based collision response becomes the bottleneck, and our integrated algorithm for adaptive simulation and collision detection shows its power at best when the bears come close to rest in a large contact group, reaching a frame rate of 12 fps in this situation.

The falling knot in Figure 5 shows the ability to robustly handle smooth contact areas, even for self-collisions. We have also evaluated the performance of collision detection, as outlined in Table 2, using a surface mesh $S = \partial M_1$ with 42254 triangles and a non-adaptive simulation mesh $M_f = M_4$ with 558 tetrahedra. Our algorithm shows an average performance of 23.6 fps. Even though this benchmark is a worst-case scenario, as large regions of the BVH are visited due to the large contact areas, we obtain 7x average speed-up over full refitting of the BVH, and 3x speed-up over spatial hashing applied to the surface S (Not to be confused with our use of spatial hashing for tetrahedra). Speedups would grow with a denser surface S, thanks to the sublinear cost of our algorithm, as well as with smaller contact regions.

A compelling application of our unified treatment of adaptive simulation and collision detection is the interactive deformation of the self-colliding dragon. The model suffers global deformations as well as detailed localized deformations due to contact with the bear. However, note that potential colliding areas are inherently large as we solve a self-collision detection problem. We used a surface mesh $S = \partial M_1$ with 13480 triangles, and a simulation mesh $M_f = M_4$ with 2552 tetrahedra. We disabled adaptive refinements to ensure that constraint-based collision response remained tractable at interactive rates. The frame rate varied from 8 to 15 fps when playing a pre-recorded bear trajectory (Performance halves when the haptic device is connected at runtime). The time per frame can be approximately decomposed into: 50 ms for the forward dynamics update, 15 ms for the intersection query of front tetrahedra through spatial hashing, 20 ms for the update of the BVH, 5 to 25 ms for the BVH self-collision query (notice the fast query due to hier-

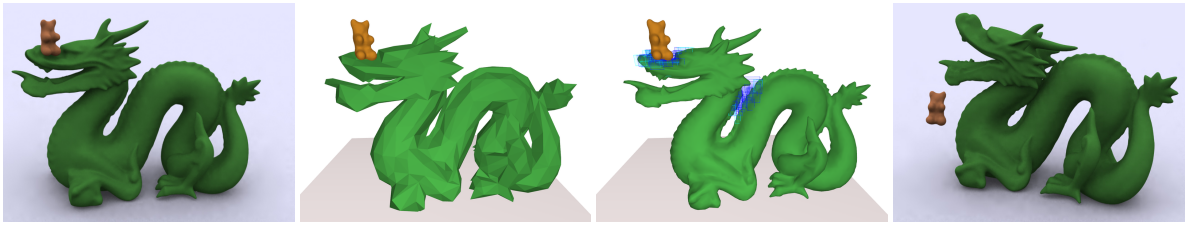


Figure 8: ***Hierarchical Pruning of Self-Collisions.*** *(i) A dragon with a surface $S = \partial M_1$ with 13 480 triangles is simulated using (ii) a mesh M_4 (with 2 552 tetrahedra). (iii) In the level L_3 of the BVH, only the MBVs depicted in blue need to be updated, as our hierarchical self-collision detection algorithm can prune regions governed by non-intersecting tetrahedra at high levels of the BVH. (iv) The self-collision between back and neck is handled accurately and effiicently.*

Num. Bears	Fw. Update (fps)			Total (fps)		
	CG	MG	Ours	CG	MG	Ours
1	15.3	17.8	383.2	3.07	3.37	58.2
8	1.89	2.15	26.7	0.40	0.36	2.98

Table 1: ***Comparison of Solvers.*** *The benchmark of Figure 4 is executed with 1 and 8 bears: (i) at full res. with conjugate gradient (CG); (ii) at full res. with a multigrid solver (MG); (iii) with our adaptive multigrid solver. Average total performance and for forward dynamics update are shown.*

Full BVH	Hash Grid	W/o k-DOP	W/o Sort	Ours		
				(1)	(2)	(3)
282.5	111.9	314.8	62.4	21.6	2.8	18.0

Table 2: ***Self-Collision Detection Timings (in ms.).*** *Our algorithm (1: MBH refit, 2: tet hashing, 3: query) Vs. (i) a full BVH-based query, (ii) spatial hashing on the full surface, (iii) no k-DOPs, and (iv) no tet sorting, on a 42 254-triangle knot simulated with 558 tetrahedra (See Figure 5).*

archical pruning), and up to 250 ms for collision response (10 ms per constraint anticipation, and 40 ms for the final response computation).

Our algorithm exploits simulation adaptivity to produce considerable speed-ups in collision detection even for moderately-sized surface meshes (tens of thousands of triangles), and speed-ups would be higher with denser surfaces. Although not shown here, our framework can also incorporate interruptible collision detection and/or perceptual error metrics. Examples like the dragon show the ability to simulate in real-time challenging self-collisions, but our framework is also applicable to offline simulations, for example for fast preview of simulations of complex embedded surfaces. One can compute a coarse simulation (and render a coarse preview surface) while conforming to the boundary conditions of the full resolution surface, therefore guaranteeing that the final rendering will be collision-free.

7. Summary and Future Work

Motivated by the observation that setting boundary conditions in previous adaptive deformable simulation methods required handling the boundary in a non-adaptive manner, we have designed data structures and algorithms for seamless integration of adaptive simulation and collision detection. In particular, our novel multiscale bounding hierarchies link mesh hierarchies employed in adaptive simulation with BVHs for collision detection. Coupled with novel adaptive multigrid simulation and collision detection methods, they enable robust and accurate handling of contact boundary conditions with a cost independent on the boundary's complexity. As an example, self-collision of complex surfaces with tens of thousands of triangles, governed by a few thousand DoFs, can be simulated interactively.

As with other hierarchical and/or adaptive techniques, performance comes with the price of memory consumption ($O(n \lg n)$ for our multiscale bounding hierarchies) and the algorithms require slightly more elaborate implementation. This was not a major issue in our case, because the collision detection builds on well-known algorithms, and our contributions in the adaptive simulation enable the use of the very same algorithms as in the past (but now on irregular meshes).

Although we have not suffered lack of convergence problems in the simulations, it is recommendable to analyze the influence of our interpolation and relaxation strategies for irregular meshes. From a performance point of view, collision response becomes the bottleneck in our simulations in situations with large contact areas. We plan to explore solutions for contact clustering that exploit adaptivity as well. And, in the future, we would also like to extend our method to handle the simulation of 2D manifolds embedded in 3D.

Acknowledgements

We would like to thank the anonymous reviewers, Sabine Coquillart, Kaspar Schüpbach, Gilles Debunne, Nico Galoppo, Remo Ziegler, Denis Steinemann, Mario Botsch, Nico Pietroni, Martin Wicke, and others at the CGL at ETH Zurich. This project was supported in part by the NCCR CoMe of the Swiss National Science Foundation.

References

[AKS05] AKSOYLU B., KHODAKOVSKY A., SCHRÖDER P.: Multilevel solvers for unstructured surface meshes. *SIAM Journal on Scientific Computing 26*, 4 (2005), 1146–1165.

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

[BHM00] BRIGGS W. L., HENSON V. E., MCCORMICK S. F.: *A Multigrid Tutorial, 2nd Edition*. SIAM, 2000.

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

[Bra77] BRANDT A.: Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation 31*, 138 (1977).

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. *Proc. of ACM SIGGRAPH* (1998).

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

[CPS92] COTTLE R., PANG J., STONE R.: *The Linear Complementarity Problem*. Academic Press, 1992.

[dB97] DEN BERGEN G. V.: Efficient collision detection of complex deformable models using aabb trees. *J. Graphics Tools 2*, 4 (1997), 1–14.

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

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

[GLM96] GOTTSCHALK S., LIN M., MANOCHA D.: OBB-Tree: A hierarchical structure for rapid interference detection. *Proc. of ACM SIGGRAPH* (1996), 171–180.

[GM97] GIBSON S. F., MIRTICH B. V.: *A Survey of Deformable Modeling in Computer Graphics*. Tech. rep., Mitsubishi Electric Research Laboratory, 1997.

[GRLM03] GOVINDARAJU N., REDON S., LIN M., MANOCHA D.: CULLIDE: Interactive collision detection between complex models in large environments using graphics hardware. *Proc. of ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware* (2003), 25–32.

[GTS02] GREEN S., TURKIYYAH G., STORTI D.: Subdivision-based multilevel methods for large scale engineering simulation of thin shells. *Proc. of ACM Symposium on Solid Modeling and Applications* (2002), 265–272.

[GW05] GEORGII J., WESTERMANN R.: A multigrid framework for real-time simulation of deformable volumes. *Proc. of VRIPHYS* (2005).

[ITF04] IRVING G., TERAN J., FEDKIW R.: Invertible finite elements for robust simulation of large deformation. *Proc. of ACM SIGGRAPH/Eurographics SCA* (2004), 131–140.

[JP03] JAMES D. L., PAI D. K.: Multiresolution Green's function methods for interactive simulation of large-scale elastostatic objects. *ACM TOG 22*, 1 (2003).

[JP04] JAMES D. L., PAI D. K.: BD-Tree: Output-sensitive collision detection for reduced deformable models. *Proc. of ACM SIGGRAPH* (2004).

[KZ05] KAVAN L., ZARA J.: Fast collision detection for skeletally deformable models. *Proc. of Eurographics* (2005).

[LAM01] LARSSON T., AKENINE-MÖLLER T.: Collision detection for continuously deforming bodies. *Eurographics* (2001).

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

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. *Proc. of ACM SIGGRAPH* (2005), 471–478.

[MT86] MCCORMICK S. F., THOMAS J.: The fast adaptive composite grid (FAC) method for elliptic equations. *Mathematics of Computation 46*, 174 (1986), 439–456.

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

[OBH02] O'BRIEN J. F., BARGTEIL A. W., HODGINS J. K.: Graphical modeling and animation of ductile fracture. *Proc. of ACM SIGGRAPH* (2002), 291–294.

[ODGK03] O'SULLIVAN C., DINGLIANA J., GIANG T., KAISER M. K.: Evaluating the visual fidelity of physically based animations. *Proc. of ACM SIGGRAPH* (2003), 527–536.

[OL03] OTADUY M. A., LIN M. C.: Sensation preserving simplification for haptic rendering. *Proc. of ACM SIGGRAPH* (2003).

[Ran99] RANNACHER R.: Error control in finite element computations. *NATO Science Series, Series C: Mathematics and Physical Sciences 536* (1999), 247–278.

[RKC02] REDON S., KHEDDAR A., COQUILLART S.: Fast continuous collision detection between rigid bodies. *Proc. of Eurographics* (2002).

[SGG*06] SUD A., GOVINDARAJU N. K., GAYLE R., KABUL I., MANOCHA D.: Fast proximity computation among multiple deformable models using discrete voronoi diagrams. *Proc. of ACM SIGGRAPH* (2006).

[SYBF06] SHI L., YU Y., BELL N., FENG W.-W.: A fast multigrid algorithm for mesh deformation. *Proc. of ACM SIGGRAPH* (2006).

[THM*03] TESCHNER M., HEIDELBERGER B., MÜELLER M., POMERANETS D., GROSS M.: Optimized spatial hashing for collision detection of deformable objects. *Proc. of Vision, Modeling and Visualization* (2003).

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

[TOS01] TROTTENBERG U., OOSTERLEE C., SCHÜLLER A.: *Multigrid*. Academic Press, 2001.

[VMT94] VOLINO P., MAGNENAT-THALMANN N.: Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. *Eurographics* (1994).

[VMT06] VOLINO P., MAGNENAT-THALMANN N.: Resolving surface collisions through intersection contour minimization. *Proc. of ACM SIGGRAPH* (2006).

[WT04] WU X., TENDICK F.: Multigrid integration for interactive deformable body simulation. *International Symposium on Medical Simulation* (2004).