Seamless Adaptivity of Elastic Models

Maxime Tournier, Matthieu Nesme, François Faure, Benjamin Gilles

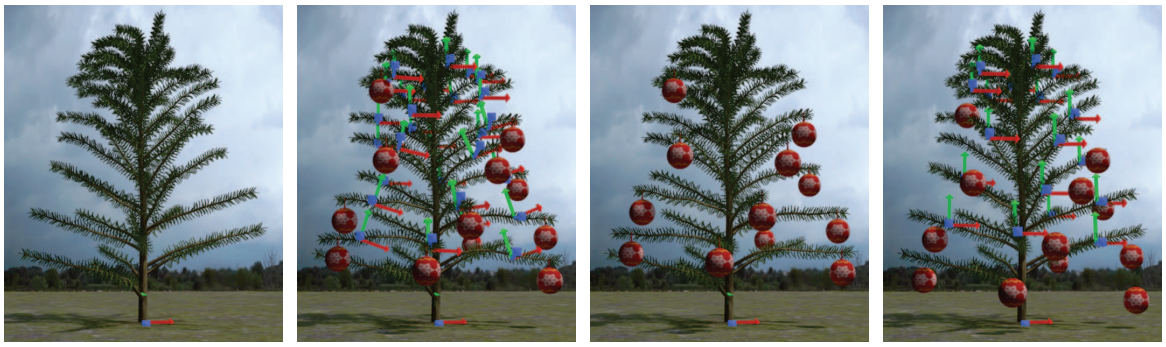
Seamless Adaptivity of Elastic Models

Maxime Tournier
INRIA
LIRMM-CNRS
Université Montpellier 2

Matthieu Nesme
INRIA
LJK-CNRS
Université de Grenoble

Francois Faure
INRIA
LJK-CNRS
Université de Grenoble

Benjamin Gilles
INRIA
LIRMM-CNRS
Université Montpellier 2



(a) Rest state (b) Compressing (c) Equilibrium (d) Decompressing

Figure 1: Deformable Christmas tree with adaptive velocity field. (1a): One frame is sufficient in steady state. (1b): When ornaments are attached, additional frames are activated to allow deformation. (1c): The velocity field can be simplified again when the equilibrium is reached. Note that our method can simplify locally deformed regions. (1d): Once the branches are released, the velocity field is refined again to allow the branches to recover their initial shape.

ABSTRACT

A new adaptive model for viscoelastic solids is presented. Unlike previous approaches, it allows seamless transitions, and simplifications in deformed states. The deformation field is generated by a set of physically animated frames. Starting from a fine set of frames and mechanical energy integration points, the model can be coarsened by attaching frames to others, and merging integration points. Since frames can be attached in arbitrary relative positions, simplifications can occur seamlessly in deformed states, without returning to the original shape, which can be recovered later after refinement. We propose a new class of velocity-based simplification criterion based on relative velocities. Integration points can be merged to reduce the computation time even more, and we show how to maintain constant elastic forces through the levels of detail. This meshless adaptivity allows significant improvements of computation time.

Index Terms: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

1 INTRODUCTION

The stunning quality of high-resolution physically based animations of deformable solids requires complex deformable models with large numbers of independent Degrees Of Freedom (DOF) which result in large dynamics equation systems and high computation times. On the other hand, the thrilling user experience provided by interactive simulations can only be achieved using fast computation times which preclude the use of high-resolution models. Reconciling these two contradictory goals requires adaptive models to efficiently manage the number of DOFs, by refining the model where necessary and coarsening it where possible. Mesh-based deformations can be seamlessly refined by subdividing elements and interpolating new nodes within these. However, seamless coarsening can be performed only when the fine nodes are back to their original position with respect to their higher-level elements, which happens only in the locally undeformed configurations (*i.e.* with null strain). Otherwise, a popping artifact (*i.e.* , an instantaneous change of shape) occurs. This not only violates the laws of Physics, but it is also visually disturbing. Simplifying objects in deformed configurations, as demonstrated in Fig. 1c, has thus not been possible with previous adaptive approaches, unless the elements are small or far enough. This may explain why extreme coarsening has rarely been proposed, and adaptive FEM models typically range from moderate to high complexity.

We introduce a new approach of adaptivity to mechanically simplify objects in arbitrarily deformed configurations, while exactly maintaining their current shape and controlling the velocity discontinuity, which we call seamless adaptivity. It extends a frame-based meshless method and straightforwardly derives from the ability of attaching frames to others in arbitrary relative positions, as illustrated in Fig. 2. In this example, a straight beam is initially animated using a single moving frame, while another control frame is attached to it. We then detach the child frame to allow the bending of the beam. If the deformation of the beam becomes constant, its *velocity* can again be modeled using a single moving frame, while its *shape* can be frozen in a deformed state by applying an offset to its reference position with respect to the active frame. Setting the offset to the current relative position removes mechanical DOFs without altering the current shape of the object. This deformation is reversible. If the external loading applied to the object changes, we can mechanically refine the model again (*i.e.* activate the passive frame) to allow the object to recover its initial shape or to undergo

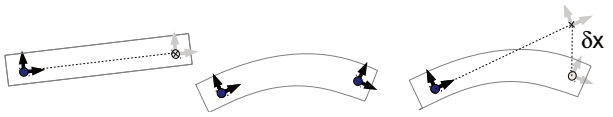


Figure 2: Seamless coarsening in a deformed state. Left: reference shape, one active frame in black, and a passive frame in grey attached using a relative transformation (dotted line). Middle: activating the frame allows it to move freely and to deform the object. Right: deactivated frame in a deformed configuration using an offset δx.

new deformations. The ability to dynamically adapt the velocity field independently of the deformation is the specific feature of our approach which dramatically enhances the opportunities for coarsening mechanical models compared with previous methods.

Our specific contributions are (1) a deformation method based on a generalized frame hierarchy for dynamically tuning the complexity of deformable solids with seamless transitions; (2) a novel simplification and refinement criterion based on velocity, which allows us to simplify the deformation model in deformed configurations, and (3) a method to dynamically adapt the integration points and enforce the continuity of forces across changes of resolution.

The remainder of this article is organized as follows. We summarize the original frame-based simulation method and we define some notations in Section 3. An overview of our adaptive framework is presented in Section 4. We discuss our strategy for nodal adaptivity in Section 5. The adaptivity of the integration points is then introduced in Section 6. We discuss results in Section 7 and future work in Section 8.

2 Related Work

The simulation of viscoelastic solids is a well-studied problem in computer graphics, starting with the early work of Terzopoulos et al. [29]. A survey can be found in [24]. Frame-based models have been proposed [20, 22, 9, 7], and the impressive efficiency of precomputed reduced models has raised a growing interest [17, 2, 3, 15, 11, 12], but run-time adaptivity remains a challenge. The remainder of this review focuses on this issue.

Hutchinson et al. [13] and Ganovelli et al. [8] first combined several resolutions of 2D and 3D solids dynamically deformed by mass-springs. Cotin et al. [5] combined two mechanical models to simulate various parts of the same object. Most adaptive methods are based on meshes at multiple resolutions. Mixing different mesh sizes can result in T-nodes that are mechanically complex to manage in the Finite Element Method (FEM). Wu et al. [32] chose a decomposition scheme that does not generate such nodes. Debunne et al. [6] performed the local explicit integration of non-nested meshes. Grinspun et al. [10] showed that hierarchical shape functions are a generic way to deal with T-nodes. Sifakis et al. [26] constrained T-nodes within other independent nodes. Martin et al. [21] solved multi-resolution junctions with polyhedral elements. Several authors proposed to generate on the fly a valid mesh with dense and fine zones. Real-time remeshing is feasible for 1D elements such as rods and wires [18, 27, 25] or 2D surfaces like cloth [23]. For 3D models, it is an elegant way to deal with cuttings, viscous effects and very thin features [4, 31, 30]. A mesh-less, octree-based adaptive extension of shape matching has been proposed [28]. Besides all these methods based on multiple resolutions, Kim and James [16] take a more algebraic approach, where the displacement field is decomposed on a small, dynamically updated, basis of orthogonal vectors, while a small set of carefully chosen integration points are used to compute the forces. In constrast to these works, our method relies on velocity field adaptation and a meshless discretization.

Numerous error estimators for refinement have been proposed in conventional FEM analysis. For static analysis, they are generally based on a precomputed stress field. This is not feasible in real time dynamics, where the current configuration must be used. Wu et al. [32] proposed four criteria based on the curvature of the stress, strain or displacement fields. Debunne et al. [6] considered the Laplacian of the displacement. Lenoir et al. [18] refined parts in contact for wire simulation. These approaches refine the objects where they are the most deformed, and they are not able to save computation time in equilibrium states. The problems relative to the criterion thresholds are rarely discussed. The smaller the thresholds, the smaller the popping artifacts, but also the more difficult to simplify thus the less efficient.

3 Frame-based Simulation Method

In this section we summarize the method that our contribution extends, and we introduce notations and basic equations. The method of [7] performs the physical simulation of viscoelastic solids using a hyperelastic formulation. The control nodes are moving frames with 12 degrees of freedom (DOF) which positions, velocities and forces in world coordinates are stored in state vectors $\mathbf{x}$, $\mathbf{v}$ and $\mathbf{f}$. The world coordinates of frame i are the entries of the 4×4 homogeneous matrix $\mathbf{X}_i$, while $\mathbf{X}_i^j$ denotes its coordinates with respect to frame j. These nodes control objects using a Skeleton Subspace Deformation (SSD) method, also called *skinning* [19]. We use Linear Blend Skinning (LBS), though other methods would be suitable (see e.g. [14] for a discussion about SSD techniques). The position of a material point i is defined using a weighted sum of affine displacements:

$$\mathbf{p}_i(t) = \sum_{j\in\mathcal{N}} \phi_i^j \mathbf{X}_j(t)\mathbf{X}_j(0)^{-1}\mathbf{p}_i(0) \qquad (1)$$

where $\mathcal{N}$ is the set of control nodes, and ϕ_i^j is the value of the shape function of node j at material position $x_i(0)$, computed at initialization time using distance ratios as in [7]. Spatially varying shape functions allow deformations. Similarly with nodes, the state of all skinned points are stored as vectors: $\mathbf{p}$, $\dot{\mathbf{p}}$, and $\mathbf{f_p}$. By differentiation of Eq. (1), a *constant* Jacobian matrix $\mathbf{J}_p$ can be assembled at initialization, relating control node and points: $\mathbf{p}=\mathbf{J}_p\mathbf{x}$, $\dot{\mathbf{p}}=\mathbf{J}_p\mathbf{v}$.

External forces can be applied directly to the nodes, or to the contact surface of the object. One can show using the Principle of Virtual Work that the skin forces $\mathbf{f_p}$ can be converted to nodal forces as: $\mathbf{f}=\mathbf{J}_p^T\mathbf{f_p}$. A generalized mass matrix for nodes can thus be assembled at initialization based on scalar masses of skinned particles $\mathbf{M}_p$: $\mathbf{M}=\mathbf{J}_p^T\mathbf{M}_p\mathbf{J}_p$. As shown in [9], differentiating Eq. (1) with respect to material coordinates allows the mapping of deformation gradients instead of points. By mapping deformation gradients to strains (such as Cauchy, Green-Lagrange or corotational), and applying a constitutive law (such as Hooke or Mooney-Rivlin), we can compute the elastic potential energy density at any location. After spatial integration and differentiation with respect to the degrees of freedom, forces can be computed and propagated back to the nodes.

We use different discretizations for visual surfaces, contact surfaces, mass and elasticity (potential energy integration points). Masses are precomputed using a dense volumetric rasterization, where voxels are seen as point masses. Deformation gradient samples (*i.e.* Gauss points) are distributed so as to minimize the numerical integration error (see Sec. 6)). For each sample, volume

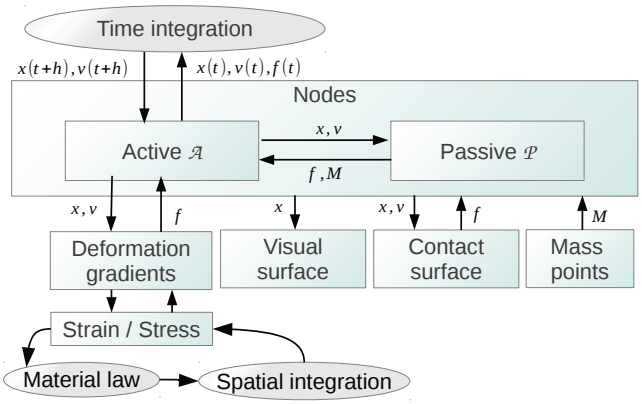


Figure 3: Kinematic structure of the simulation. Our adaptive scheme splits the control nodes into active (*i.e.* independent) nodes and passive (*i.e.* mapped) nodes.

moments are precomputed from the fine voxel grid and associated with local material properties.

The method is agnostic with respect to the way we solve the equations of motion. We apply an implicit time integration to maintain stability in case of high stiffness or large time steps [1]. At each time step, we solve a linear equation system

$$\mathbf{A}\Delta\mathbf{v} = \mathbf{b} \tag{2}$$

where $\Delta\mathbf{v}$ is the velocity change during the time step, matrix $\mathbf{A}$ is a weighted sum of the mass and stiffness matrices, while the right-hand term depends on the forces and velocities at the beginning of the time step. The main part of the computation time to set up the equation system is proportional to the number integration points, while the time necessary to solve it is a polynomial function of the number of nodes (note that $\mathbf{A}$ is a sparse, positive-definite symmetric matrix).

4 Adaptive Frame-based Simulation

Our first extension to the method presented in Sec. 3) is to attach some control nodes to others to reduce the number of independent DOFs and deformation modes. This amounts to adding an extra block to the kinematic structure of the model, as shown in Fig. 3. The independent state vectors are restricted to the active nodes. At each time step, the dynamics equation is solved to update the positions and velocities of the active nodes, then the changes are propagated to the passive nodes, then to the skin points and the material integration points. The forces are propagated the other way round. When a node i is passive, its matrix is computed using LBS as

$$\mathbf{X}_i(t) = \sum_{j\in\mathscr{A}} \phi_i^j \mathbf{X}_j(t)\mathbf{X}_j(0)^{-1}\mathbf{X}_i(0) \tag{3}$$

where $\mathscr{A}$ is the set of active nodes and ϕ_i^j is the value of the shape function of node j at the origin of $\mathbf{X}_i$ in the reference, undeformed configuration. The point positions of Eq. (1) can be written in terms of active nodes only:

$$\mathbf{p}_i(t) = \sum_{j\in\mathscr{A}} \psi_i^j \mathbf{X}_j(t)\mathbf{X}_j(0)^{-1}\mathbf{p}_i(0) \tag{4}$$

$$\psi_i^j = \phi_i^j + \sum_{k\in\mathscr{P}} \phi_k^j \phi_i^k \tag{5}$$

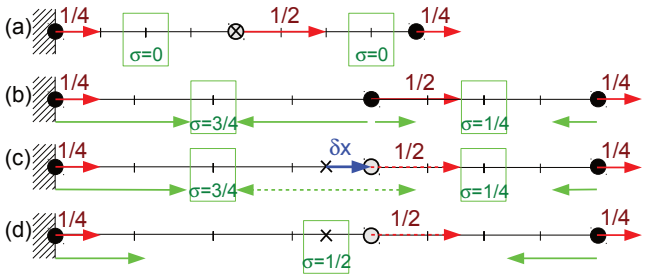


Figure 4: Refinement and simplification. Red and green arrows denote external and internal forces, respectively. Plain circles represent active nodes, while empty circles represent passive nodes attached to their parents, and crosses represent the positions of passive nodes interpolated from their parents positions. Dashed lines are used to denote forces divided up among the parent nodes. Rectangles denote integration points, where the stresses σ are computed. (a): A bar in reference state undergoes external forces and starts stretching. (b): In rest state, 3 active nodes. (c): With the middle node attached with an offset with respect to the interpolated position. (d): After replacing two integration points with one.

where $\mathscr{P}$ is the set of passive nodes. These equations straightforwardly generalize to deformation gradients. This easy composition of LBS is exploited in our node hierarchy (Sec. 5.2)) and our adaptive spatial integration scheme (Sec. 6)). At any time, an active node i can become passive. Since the coefficients used in Eq. (3) are computed in the undeformed configuration, the position $\bar{\mathbf{X}}_i$ computed using this equation is different from the current position $\mathbf{X}_i$, and moving the frame to this position would generate an artificial instantaneous displacement. To avoid this, we compute the offset $\delta\mathbf{X}_i = \bar{\mathbf{X}}_i^{-1}\mathbf{X}_i$, as illustrated in Fig. 2. The skinning of the frame is then biased by this offset as long as the frame remains passive, and its velocity is computed using the corresponding Jacobian matrix:

$$\mathbf{X}_i(t) = \sum_{j\in\mathscr{A}} \psi_i^j \mathbf{X}_j(t)\mathbf{X}_j(0)^{-1}\mathbf{X}_i(0)\delta\mathbf{X}_i \tag{6}$$

$$\mathbf{u}_i(t) = \mathbf{J}_i\mathbf{v}(t) \tag{7}$$

Our adaptivity criterion is based on the comparison of the velocity of a passive node attached to nodes of $\mathscr{A}$, with the velocity of the same node moving independently; if the difference is below a threshold the node should be passive, otherwise it should be active.

One-dimensional Example

A simple one-dimensional example is illustrated in Fig. 4. A bar is discretized using three control nodes and two integration points, and stretched horizontally by its weight, which applies the external forces $1/4$, $1/2$ and $1/4$, from left to right respectively. For simplicity we assume unitary gravity, stiffness and bar section, so that net forces are computed by simply summing up strain and force magnitudes. At the beginning of the simulation, Fig. 4a, the bar is in reference configuration with null stress, and the middle node is attached to the end nodes, interpolated between the two. The left node is fixed, the acceleration of the right node is 1, and the acceleration of the interpolated node is thus $1/2$. However, the acceleration of the corresponding *active* node would be 1, because with null stress, it is subject to gravity only. Due to this difference, we activate it and the bar eventually converges to the equilibrium configuration shown in Fig. 4b, with a non-uniform extension, as can be visualized using the vertical lines regularly spaced in the material domain.

Once the center node is stable with respect to its parents, we can simplify the model by attaching it to them, with offset δx. External and internal forces applied to the passive node, which balance each other, are divided up among its parents, which do not change the net force applied to the end node. The equilibrium is thus maintained. The computation time is faster since there are less unknown in the dynamics equation. However, computing the right-hand term remains expensive since the same two integration points are used.

Once the displacement field is simplified, any change of strain due to the displacements of the two independent nodes is uniform across the bar. We thus merge the two integration points to save computation time, as shown in Fig. 4d.

Section 5 details node adaptivity, while the adaptivity of integration points is presented in Section 6.

5 Adaptive Kinematics

5.1 Adaptivity Criterion

At each time step, our method partitions the nodes into two sets: the *active* nodes, denoted by $\mathscr{A}$, are the currently independent DOFs from which the *passive* nodes, denoted by $\mathscr{P}$, are mapped. We further define a subset $\mathscr{AC} \subset \mathscr{P}$ to be composed of nodes candidate for activation. Likewise, the deactivation candidate set is a subset $\mathscr{PC} \subset \mathscr{A}$. To decide whether candidate nodes should become passive or active, we compare their velocities in the two cases and change their status when the velocity difference crosses a certain user-defined threshold η discussed below. At each time step, we thus compare the velocities in the three following cases:

1. with $\mathscr{A} \setminus \mathscr{PC}$ active and $\mathscr{P} \cup \mathscr{PC}$ passive (coarser resolution)
2. with $\mathscr{A}$ active and $\mathscr{P}$ passive (current resolution)
3. with $\mathscr{A} \cup \mathscr{AC}$ active and $\mathscr{P} \setminus \mathscr{AC}$ passive (finer resolution)

We avoid solving the three implicit integrations, noticing that cases 1 and 3 are only used to compute the adaptivity criterion. Instead of performing the implicit integration for case 1, we use the solution given by 2 and we compute the velocities of the frames in $\mathscr{PC}$ as if they were passive, using Eq. (7). For case 3, we simply use an explicit integration for the additional nodes $\mathscr{AC}$, in linear time using a lumped mass matrix. In practice, we only noticed small differences with a fully implicit integration. At worse, overshooting due to explicit integration temporary activates too many nodes.

Once every velocity difference has been computed and measured for candidate nodes, we integrate the dynamics forward at current resolution (*i.e.* using system 2), then we update the sets $\mathscr{A}, \mathscr{P}, \mathscr{PC}, \mathscr{AC}$ and finally move on to the next time step.

Metrics

For a candidate node i, the difference between its passive and active velocities is defined as:

$$\mathbf{d}_i = \mathbf{J}_i(\mathbf{v} + \Delta\mathbf{v}) - (\mathbf{u}_i + \Delta\mathbf{u}_i) \tag{8}$$

where $\mathbf{J}_i$ is the Jacobian of Eq. (7), and $\Delta\mathbf{v}, \Delta\mathbf{u}_i$ are the velocity updates computed by time integration, respectively in the case where the candidate node is passive and active. Note that for the activation criterion computed using explicit integration (case 3), this reduces to the generalized velocity difference $\mathbf{d}_i = \mathbf{J}_i\Delta\mathbf{v} - dt\tilde{\mathbf{M}}_i^{-1}\mathbf{f}_i$ where $\tilde{\mathbf{M}}_i$ is the lumped mass matrix block of node i, $\mathbf{f}_i$ its net external force and dt is the time step , which is a difference in *acceleration* up to dt. A measure of $\mathbf{d}_i$ is be computed as:

$$\mu_i = ||\mathbf{d}_i||^2_{\mathbf{W}_i} := \frac{1}{2}\mathbf{d}_i^T\mathbf{W}_i\mathbf{d}_i \tag{9}$$

where $\mathbf{W}_i$ is a positive-definite symmetric matrix defining the metric (some specific $\mathbf{W}_i$ are shown below). The deactivation (respectively activation) of a candidate node i occurs whenever $\mu_i \leq \eta$ (respectively $\mu_i > \eta$), where η is a positive user-defined threshold.

Kinetic Energy As the nodes are transitioning between passive and active states, a velocity discontinuity may occur. In order to prevent instabilities, a natural approach is to bound the associated kinetic energy discontinuity. We do so using $\mathbf{W}_i = \mathbf{M}$ in Eq. (9). The total kinetic energy difference introduced by changing k candidate node states is:

$$\mu_{total} = \Big|\Big|\sum_i^k \mathbf{d}_i\Big|\Big|^2_{\mathbf{M}} \leq \sum_i^k ||\mathbf{d}_i||^2_{\mathbf{M}} = \sum_i^k \mu_i \tag{10}$$

Thus, placing a threshold on each individual μ_i effectively bounds the total kinetic energy discontinuity. The criterion threshold η can then be adapted so that the upper bound in Eq. (10) becomes a small fraction of the current kinetic energy.

Distance to Camera For Computer Graphics applications, one is usually ready to sacrifice precision for speed as long as the approximation is not visible to the user. To this end, we can measure velocity differences according to the distance to the camera of the associated visual mesh, so that motion happening far from the camera will produce lower measures, thus favoring deactivation. More precisely, if we call $\mathbf{G}_i$ the kinematic mapping between node i and the mesh vertices, and $\mathbf{Z}$ a diagonal matrix with positive values decreasing along with the distance between mesh vertices and the camera, the criterion metric is then given by:

$$\mathbf{W}_i = \mathbf{G}_i^T\mathbf{Z}\mathbf{G}_i \tag{11}$$

In practice, we use a decreasing exponential for $\mathbf{Z}$ values (1 on the camera near-plane, 0 on the camera far-plane) in the spirit of the decreasing precision found in the depth buffer during rendering. The two metrics can also be combined by retaining the minimum of their values: simplification is then favored far from the camera, where the distance metric is always small, while the kinetic energy metric is used close to the camera, where the distance metric is always large.

5.2 Adaptive Hierarchy

In principle, we could start with an unstructured fine node discretization of the objects and at each time step, find the best simplifications by considering all possible deactivation and activation candidates. To avoid a quadratic number of tests, we pre-compute a node hierarchy and define candidate nodes to be the ones at the front between passive and active nodes.

Hierarchy Setup Our hierarchy is computed at initialization time, as illustrated in Fig. 5. At each level, we perform a Lloyd relaxation on a fine voxel grid to spread new control nodes as evenly as possible, taking into account the frames already created at coarser levels. Before updating the shape functions, we interpolate the weights ϕ_j^i at the origin of each new node j, relative to the nodes i at coarser levels. For each non-null weight, an edge is inserted in the dependency graph, resulting in a generalized hierarchy based on a Directed Acyclic Graph.

Hierarchy Update The candidates for activation are the passive nodes with all parents active. Conversely, the candidates for deactivation are the active nodes with all children passive, except the root

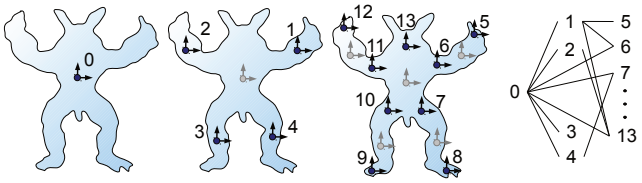


Figure 5: Reference node hierarchy. From left to right: the first three levels, and the dependency graph.

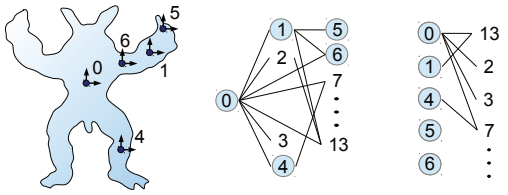


Figure 6: Mechanical hierarchy. Left: active nodes $0,1,4,5,6$ at a given time. Middle: Reference hierarchy; nodes $2,3,7$ are activation candidates; nodes $4,5,6$ are deactivation candidates. Right: the resulting two-level contracted graph to be used in the mechanical simulation.

of the reference hierarchy. In the example shown in Fig. 6, nodes $4,5,6,1$, and 0 shown in the character outline are active. As such, they do not mechanically depend on their parents in the reference hierarchy, and the mechanical dependency graph is obtained by removing the corresponding edges from the reference hierarchy. For edges in this two-levels graph, weights are obtained by contracting the reference hierarchy using Eq. (4) and similarly for the different sets of passive/active nodes discussed in the beginning of this section

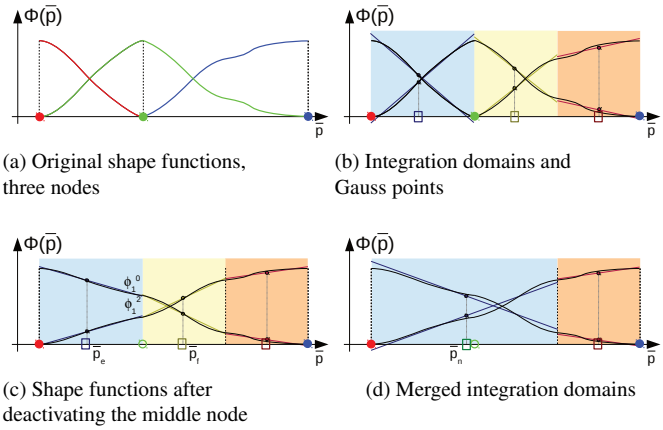
6 Adaptive Spatial Integration

6.1 Discretization

The spatial integration of energy and forces is numerically computed using Gaussian quadrature, a weighted sum of values computed at integration points. Exact quadrature rules are only available for polyhedral domains with polynomial shape functions (e.g. tri-linear hexahedra). In meshless simulation, such rules do not exist in general. However, in linear blend skinning one can easily show that the deformation gradient is uniform (respectively linear) in regions where the shape functions are constant (respectively linear). As studied in [7], uniform shape functions can be only obtained with one node, so linear shape functions between nodes are the best choice for homogeneous parts of the material, since the interpolation then corresponds to the solution of static equilibrium. One integration point of a certain degree (*i.e.* one elaston [20]) is sufficient to exactly integrate polynomial functions of the deformation gradient there, such as deformation energy in linear tetrahedra. We leverage this property to optimize our distribution of integration points. In a region $\mathscr{V}e$ centered on point $\bar{\mathbf{p}}_e$, the integral of a function g is given by:

$$\int_{\bar{\mathbf{p}}\in\mathscr{V}e} g \approx \mathbf{g}^T \int_{\bar{\mathbf{p}}\in\mathscr{V}e} (\bar{\mathbf{p}}-\bar{\mathbf{p}}_e)^{(n)} = \mathbf{g}^T \bar{\mathbf{g}}_e \tag{12}$$

where $\mathbf{g}$ is a vector containing g and its spatial derivatives up to degree n evaluated at $\bar{\mathbf{p}}_e$, while $\mathbf{p}^{(n)}$ denotes a vector of polynomials of degree n in the coordinates of $\mathbf{p}$, and $\bar{\mathbf{g}}_e$ is a vector of polynomials



(a) Original shape functions, three nodes

(b) Integration domains and Gauss points

(c) Shape functions after deactivating the middle node

(d) Merged integration domains

Figure 7: Adaptive integration points in 1D. Disks denote control nodes while rectangles denote integration points.

integrated across $\mathscr{V}e$ which can be computed at initialization time by looping over the voxels of an arbitrarily fine rasterization. The approximation of Eq. (12) is exact if n is the polynomial degree of g. Due to a possibly large number of polynomial factors, we limit our approximation to quartic functions with respect to material coordinates, corresponding to strain energies and forces when shape functions are linear and the strain measure quadratic (*i.e.* Green-Lagrangian strain).

Since the integration error is related to the linearity of shape functions, we decompose the objects into regions of as linear as possible shape functions at initial time, as shown in Fig. 7a and Fig. 7b. We compute the regions influenced by the same set of independent nodes, and we recursively split these regions until a given linearity threshold is reached, based on the error of a least squares linear fit of the shape functions. Let $\phi_i(\bar{\mathbf{p}})$ be the shape function of node i as defined in Eq. (1), and $\mathbf{c}_i^{eT}\bar{\mathbf{p}}^{(1)}$ its first order polynomial approximation in $\mathscr{V}e$. The linearity error is given by:

$$\begin{aligned}
\varepsilon(\mathbf{c}) &= \int_{\mathscr{V}e} (\phi_i(\bar{\mathbf{p}}) - \mathbf{c}^T\bar{\mathbf{p}}^{(1)})^2 && (13)\\
&= \mathbf{c}^T A^e \mathbf{c} - 2\mathbf{c}^T B_i^e + C_i^e && (14)\\
\text{with: } A^e &= \int_{\mathscr{V}e} \bar{\mathbf{p}}^{(1)}\bar{\mathbf{p}}^{(1)T} && (15)\\
B_i^e &= \int_{\mathscr{V}e} \phi_i(\bar{\mathbf{p}})\bar{\mathbf{p}}^{(1)} && (16)\\
C_i^e &= \int_{\mathscr{V}e} \phi_i(\bar{\mathbf{p}})^2 && (17)
\end{aligned}$$

We solve for the best least squares coefficients $\mathbf{c}_i^e$ minimizing ε: $\mathbf{c}_i^e = (A^e)^{-1}B_i^e$. The region with largest error is split in two until the target number of integration points or an upper bound on the error is reached.

6.2 Merging Integration Points

At run-time, the shape functions of the passive nodes can be expressed as linear combinations of the shape functions of the active nodes using Eq. (5). This allows us to merge integration points sharing the same set of active nodes (in $\mathscr{A}\cup\mathscr{AC}$), as shown in Fig. 7c. One can show that the linearity error in the union of regions e and f is given by:

$$\varepsilon = \sum_i (C_i^e + C_i^f) - \sum_i (B_i^e + B_i^f)^T (A^e + A^f)^{-1} \sum_i (B_i^e + B_i^f)$$

If this error is below a certain threshold, we can merge the integration points. The new values of the shape function (at origin) and its derivatives are: $\mathbf{c}_i^n = (A^e + A^f)^{-1}(B_i^e + B_i^f)$. For numerical precision, the integration of Eq. (12) is centered on $\bar{\mathbf{p}}_e$. When merging e and f, we displace the precomputed integrals $\bar{\mathbf{g}}_e$ and $\bar{\mathbf{g}}_f$ to a central position $\bar{\mathbf{p}}_n = (\bar{\mathbf{p}}_e + \bar{\mathbf{p}}_f)/2$ using simple closed form polynomial expansions. Merging is fast because the volume integrals of the new integration points are directly computed based on those of the old ones, without integration across the voxels of the object volume. Splitting occurs when the children are not influenced by the same set of independent nodes, due to a release of passive nodes. To speed up the adaptivity process, we store the merging history in a graph, and dynamically update the graph (instead of restarting from the finest resolution). Only the leaves of the graph are considered in the dynamics equation.

When curvature creates different local orientations at the integration points, or when material laws are nonlinear, there may be a small difference between the net forces computed using the fine or the coarse integration points. Also, since Eq. (4) only applies when rest states are considered, position offsets $\delta\mathbf{X}$ on passive nodes create forces that are not taken into account by coarse integration points. To maintain the force consistency between the different levels of details, we compute the difference between the net forces applied by the coarse integration points and the ones before adaptation. This force offset is associated with the integration point and it is added to the elastic force it applies to the nodes. Since net internal forces over the whole object are necessarily null, so is the difference of the net forces computed using different integration points, thus this force offset influences the shape of the object but not its global trajectory. In three dimension, to maintain the force offset consistent with object rotations, we project it from the basis of the deformation gradient at the integration point to world coordinates.

7 Results

7.1 Validation

To measure the accuracy of our method, we performed some standard tests on homogeneous Hookean beams under extension and flexion (see Fig. 8). We obtain the same static equilibrium solutions using standard tetrahedral finite elements and frame-based models (with/without kinematics/integration point adaptation). In extension, when inertial forces are negligible (low masses or static solving or high damping), our adaptive model is not refined as expected from the analytic solution (one frame and one integration point are sufficient). In bending, adaptivity is necessary to model non-linear variations of the deformation gradient. At equilibrium, our model is simplified as expected. Fig. 9 shows the variation of

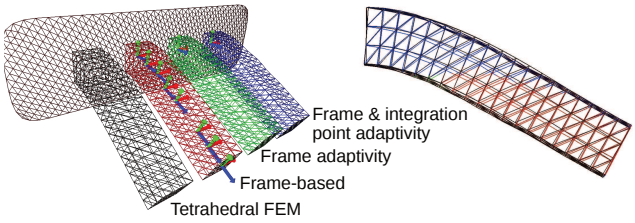
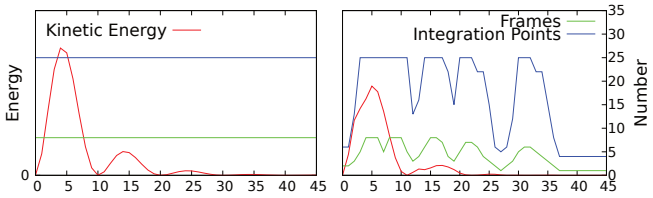


Figure 8: Four cantilever beams at equilibrium with the same properties and loading (fixed on one side and subject to gravity).

the kinetic energy (red curves). As expected, energy discontinuities remain lower than the criterion threshold when adapting nodes and integration points (green and blue curves), allowing the user to control maximum jumps in velocity. Because there is also no position discontinuities (no popping) as guaranteed by construction, the adaptive simulation in visually very close to the non-adaptive one.



(a) No Adaptivity (b) Adaptivity

Figure 9: Kinetic Energy (red) Analysis with varying number of frames (green) and integration points (blue) over time (cantilever beam under flexion).

7.2 Complex Scenes

We demonstrate the genericity of our method through the following example scenes:

Christmas Tree A Christmas tree (Fig. 1) with heterogeneous material and rigid ornaments is subject to gravity. Initially, only one node is used to represent the tree. As the ornament falls, the branches bend and nodes are automatically active until the static equilibrium is reached and the nodes become passive again. The final, bent configuration is again represented using only one control node.

Elephant Seal A simple animation skeleton is converted to control nodes to animate an elephant seal (Fig. 10) using key-frames. Adaptive, secondary motions are automatically handled by our method as more nodes are added into the hierarchy.

Figure 10: 40 adaptive, elastic frames (green=active, red=passive) adding secondary motion on a (on purpose short) kinematic skeleton corresponding to 12 (blue) frames.

Bouncing Ball A ball is bouncing on the floor with unilateral contacts (Fig. 11). As the ball falls, only one node is needed to animate it. On impact, contact constraint forces produce deformations and the nodes are active accordingly. On its way up, the ball recovers its rest state and the nodes are passive again. This shows that our method allows simplifications in non-equilibrium states.

Elastic Mushroom Field In Fig. 12a, simplification allows all the mushrooms to be attached to one single control frame until a shoe crushes some of them. Local nodes are then activated to respond to shoe contacts or to secondary contacts. They are deactivated when the shoe goes away.

Deformable Ball Stack Eight deformable balls (Fig. 13) are dropped into a glass. From left to right: (a) A unique node is necessary to simulate all balls falling under gravity, at the same speed. (b) While colliding, nodes are activated to simulate deformations.

Scene	Timing including collisions	#Steps (dt)	#Frames				#Integration Points				Speedup including collisions
			total	min	max	mean	total	min	max	mean	
Christmas Tree (Fig. 1)	5-270 ms/frame	380 (0.04s)	36	1	31	9	124	124	124	124	×1.5
Cantilever Beam (Fig. 8)	<1-110 ms/frame	370 (0.5s)	15	1	15	1.8	164	3	164	20	×2
Mushroom Field (Fig. 12a)	75-200 ms/frame	200 (0.1s)	156	1	11	5.4	251	78	88	84	×2.1
Armadillo Salad (Fig. 12b)	650-1,200 ms/frame	1,556 (0.01s)	1,800	18	1,784	365	27,162	108	27,086	5,011	×3
Ball Stack (Fig. 13)	100-250 ms/frame	20 (0.1s)	50	1	38	11.5	407	70	360	178	×1.7

Table 1: Adaptivity performances and timings.

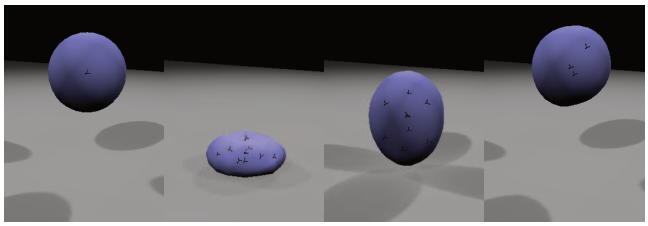
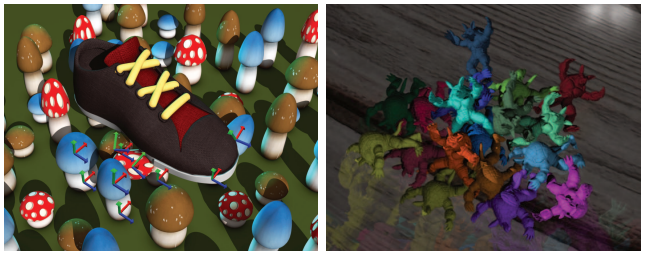


Figure 11: A falling deformable ball with unilateral contacts.



(a) Crushing elastic mushrooms (b) 18 Armadillos falling in a bowl

Figure 12: Selected pictures of complex scenes where only a subset of the available frames and integration points are active.

(c) Once stabilized, the deformed balls are simplified to one node. (d) Removing the glass, some nodes are re-activated to allow the balls to fall apart. (e) Once the balls are separated they are freely falling with air damping, and one node is sufficient to simulate all of them.

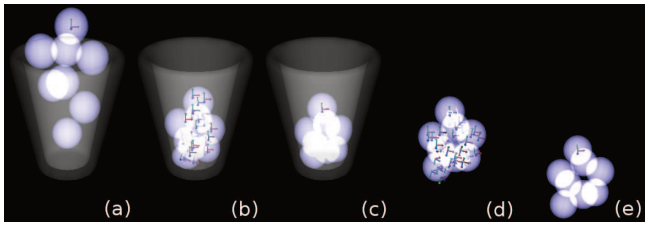


Figure 13: Eight deformable balls stacking up in a glass, which is eventually removed.

Armadillo Salad A set of Armadillos (Fig. 12b) is dropped into a bowl, demonstrating the scalability and robustness of our method in a difficult (self-)contacting situation.

7.3 Performance

In the various scenarios described above, our technique allows a significant reduction of both kinematic DOFs and integration points, as presented in Table 1. Speedups are substantial, even when collision handling is time consuming. It is worth noting that, for a fair comparison with the non-adaptive case, our examples exhibit large, global and dynamical deformations.

In order to evaluate the gain of adaptivity regarding the scene complexity, we throw armadillos in a bowl, at various resolutions. The speedups presented in Table 2 show that scenes resulting in larger systems give better speedups since the complexity of solving the system increases along with the number of DOFs. The algorithmic complexity of solving deformable object dynamics generally depends on three factors: the number of DOFs, the computation of elastic forces and, in the case of iterative solvers, the conditioning of the system. By using fewer integration points, our method is able to compute elastic forces in a much faster way. In the case of badly conditioned systems, as for instance tightly mechanically coupled system (*e.g.* stacks), iterative methods need a large number of iterations and thus the number of DOFs becomes critical. The dependency on the number of DOFs is even larger when using direct solvers. Thus, our method is particularly interesting in such cases and allow for significant speedups compared to the non-adaptive case. For instance: 6.25 when the balls are stacked into the glass (see Fig. 13c).

We noticed that the overhead due to adaptivity is moderate compared to the overall computational time (typically between 5% and 10%), since adaptivity is incremental for both nodes and integration points between two consecutive time steps. The dense voxel grid is visited only once at initialization to compute shape functions, masses, and integration data. Note that the cost of our adaptivity scheme is independent from the method to compute shape functions (they could be based on harmonic coordinates, natural neighbor interpolants, etc.).

Nb Armadillos	Max Nodes / Integration Points per Armadillo		
	10 / 49	100 / 1509	250 / 3953
1	×1.75	×3.3	×12
18	×1.5	×3	×3.1

Table 2: Speedups for a salad of one and 18 armadillos at various maximal resolutions (including collision timing)

8 Conclusion and Perspectives

We introduced a novel method for the run-time adaptivity of elastic models. Our method requires few pre-processing (few seconds) contrary to existing model reduction techniques based on modal analysis and system training. Nodes are simplified as soon as their velocities can be described by nodes at coarser levels of details, otherwise they are made independent. Linear interpolation is particularly suited for linear materials and affine deformations as it provides the static solution; therefore no refinement occurs except if inertia produces large velocity gradients. In non-linear deformation such as bending and twisting, new nodes are active to approximate the solution in terms of velocity. Using frames as kinematic primitives allows simplifications in deformed configurations based on local coordinates, which is not possible in traditional Finite Element

or particle-based techniques. Various distance metrics can be easily implemented to tune the adaptivity criterion depending on the simulation context (*e.g.* physical, visual precision). Reducing the number of independent DOFs speeds up the simulation, although the factor depends on the choice of the solver (*e.g.* iterative/direct solver, collision response method), and on the simulation scenario (*e.g.* presence of steady states, local/global, linear/non-linear deformations, mass distributions). In addition to kinematical adaptivity, we presented a method to merge integration points to speed up the computations even more of elastic internal forces. Force offsets are used to remove discontinuities between the levels of detail.

In future work, we will address the question of stiffness discontinuities and the design of scenario-dependent frame hierarchies.

Acknowledgements

The authors would like to thank Laura Paiardini for her much appreciated artistic work (modeling, rendering) and the Sofa team (http://www.sofa-framework.org) for the great simulation library. This work was partly funded by the French ANR SoHusim.

References

[1] D. Baraff and A. Witkin. Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, pages 43–54. ACM, 1998.

[2] J. Barbič and D. L. James. Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 24(3):982–990, Aug. 2005.

[3] J. Barbič and Y. Zhao. Real-time large-deformation substructuring. *ACM Trans. on Graphics (Proc. SIGGRAPH)*, 30(4):91:1–91:7, 2011.

[4] A. W. Bargteil, C. Wojtan, J. K. Hodgins, and G. Turk. A finite element method for animating large viscoplastic flow. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, volume 26, 2007.

[5] S. Cotin, H. Delingette, and N. Ayache. A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training and simulation. In *The Visual Computer*, volume 16, 2000.

[6] G. Debunne, M. Desbrun, M.-P. Cani, and A. H. Barr. Dynamic real-time deformations using space and time adaptive sampling. In *Proc. ACM SIGGRAPH*, 2001.

[7] F. Faure, B. Gilles, G. Bousquet, and D. K. Pai. Sparse Meshless Models of Complex Deformable Solids. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, volume 30, 2011.

[8] F. Ganovelli, P. Cignoni, and R. Scopigno. Introducing multiresolution representation in deformable object modeling. In *ACM Spring Conference on Computer Graphics*, 1999.

[9] B. Gilles, G. Bousquet, F. Faure, and D. Pai. Frame-based Elastic Models. In *ACM Transactions on Graphics*, volume 30, 2011.

[10] E. Grinspun, P. Krysl, and P. Schröder. Charms: a simple framework for adaptive simulation. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, volume 21, 2002.

[11] F. Hahn, S. Martin, B. Thomaszewski, R. Sumner, S. Coros, and M. Gross. Rig-space physics. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, volume 31, 2012.

[12] K. Hildebrandt, C. Schulz, C. von Tycowicz, and K. Polthier. Interactive spacetime control of deformable objects. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, volume 31, 2012.

[13] D. Hutchinson, M. Preston, and T. Hewitt. Adaptive refinement for mass/spring simulations. In *Eurographics Workshop on Computer Animation and Simulation*, pages 31–45, 1996.

[14] L. Kavan, S. Collins, J. Žára, and C. O'Sullivan. Skinning with dual quaternions. In *Proceedings of the ACM symposium on Interactive 3D graphics and games*, 2007.

[15] J. Kim and N. S. Pollard. Fast simulation of skeleton-driven deformable body characters. *ACM Transactions on Graphics*, 30, 2011.

[16] T. Kim and D. L. James. Skipping steps in deformable simulation with online model reduction. In *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, volume 28, 2009.

[17] P. G. Kry, D. L. James, and D. K. Pai. Eigenskin: real time large deformation character skinning in hardware. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer animation*, pages 153–159, 2002.

[18] J. Lenoir, L. Grisoni, C. Chaillou, and P. Meseure. Adaptive resolution of 1d mechanical b-spline. In *Proc. ACM GRAPHITE*, 2005.

[19] N. Magnenat-Thalmann, R. Laperrière, and D. Thalmann. Joint dependent local deformations for hand animation and object grasping. In *Graphics interface*, pages 26–33, 1988.

[20] S. Martin, P. Kaufmann, M. Botsch, E. Grinspun, and M. Gross. Unified simulation of elastic rods, shells, and solids. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, volume 29, 2010.

[21] S. Martin, P. Kaufmann, M. Botsch, M. Wicke, and M. Gross. Polyhedral finite elements using harmonic basis functions. In *Proc. Eurographics Symposium on Geometry Processing*, 2008.

[22] M. Müller and N. Chentanez. Solid simulation with oriented particles. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, volume 30, 2011.

[23] R. Narain, A. Samii, and J. F. O'Brien. Adaptive anisotropic remeshing for cloth simulation. In *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, volume 31, 2012.

[24] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson. Physically based deformable models in computer graphics. In *Computer Graphics Forum*, volume 25, 2006.

[25] M. Servin, C. Lacoursière, F. Nordfelth, and K. Bodin. Hybrid, multiresolution wires with massless frictional contacts. In *IEEE Transactions on Visualization and Computer Graphics*, volume 17, 2011.

[26] E. Sifakis, T. Shinar, G. Irving, and R. Fedkiw. Hybrid simulation of deformable solids. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2007.

[27] J. Spillmann and M. Teschner. An adaptive contact model for the robust simulation of knots. In *Computer Graphics Forum (Proc. Eurographics)*, volume 27, 2008.

[28] D. Steinemann, M. A. Otaduy, and M. Gross. Fast adaptive shape matching deformations. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2008.

[29] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *Proc. ACM SIGGRAPH*, 1987.

[30] M. Wicke, D. Ritchie, B. M. Klingner, S. Burke, J. R. Shewchuk, and J. F. O'Brien. Dynamic local remeshing for elastoplastic simulation. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, volume 29, 2010.

[31] C. Wojtan and G. Turk. Fast viscoelastic behavior with thin features. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, volume 27, 2008.

[32] X. Wu, M. S. Downes, T. Goktekin, and F. Tendick. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. In *Computer Graphics Forum (Proc. Eurographics)*, volume 20, 2001.