Velocity-based Adaptivity of Deformable Models

Maxime Tournier[1], Matthieu Nesme[2], Francois Faure[2], Benjamin Gilles[1]

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 continuous elastic forces through the levels of detail. Such meshless adaptivity allows significant improvements of computation time during simulations. It also provides a natural approach to coarse-to-fine deformable mesh registration.

Keywords: Computer Animation, Physically-based Animation, Deformable Solids, Adaptive Kinematics

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 equation systems for solving dynamics, 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 by 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 only happens in the locally undeformed configurations (*i.e.* with null strain). Otherwise, a popping artifact (*i.e.* an instantaneous change of shape) occurs, which not only violates the laws of physics, but it is also visually disturbing for the user. Simplifying objects in deformed configurations, as presented in Fig. 1c, has thus not been possible with previous adaptive approaches, unless the elements are small or far enough from the user. 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 naturally exploits the ability to attach 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 beam to bend as needed. If the beam deformation reaches a steady state, the *velocity* field can again be obtained from the moving frame alone, and the *shape* can be frozen in the deformed state by applying an offset to the child frame reference position relative to the moving 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 new deformations. The ability to dynamically adapt the deformation field even in non-rest configuration is the specific feature of our approach, which dramatically enhances the opportunities for coarsening mechanical models compared with previous methods.

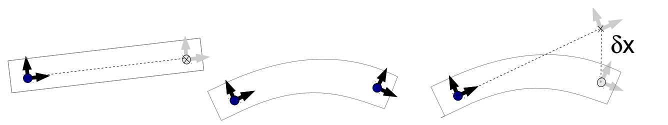
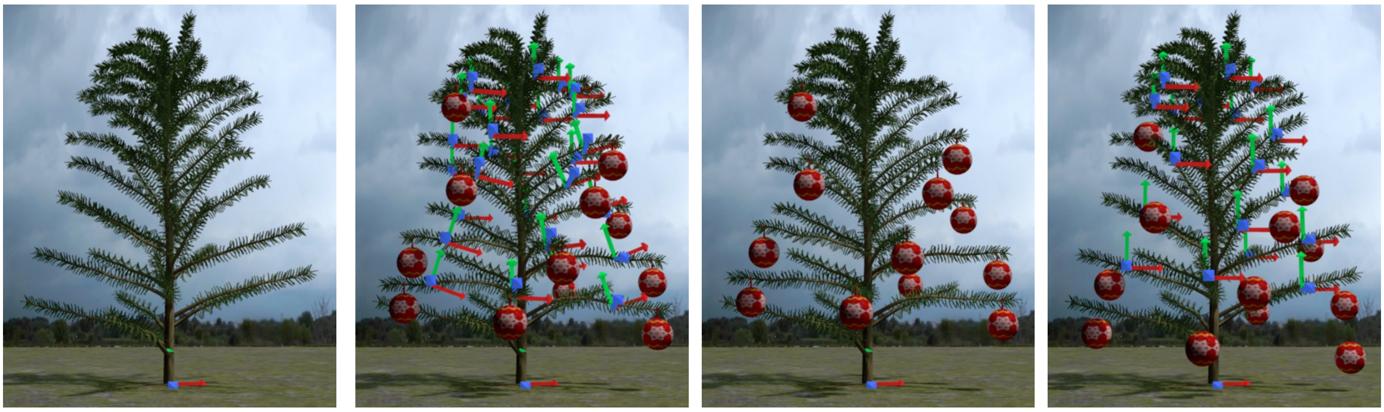


Figure 2: Seamless coarsening in a deformed state. Left: reference shape, one active frame in black, and a passive frame in grey attached using a relative transformation (dotted line). Middle: activating the frame let it to move freely and deform the object. Right: deactivated frame in a deformed configuration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

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.

[1]INRIA, LIRMM-CNRS, Université de Montpellier 2
[2]INRIA, LJK-CNRS, Université de Grenoble



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

Figure 1: Deformable Christmas tree with our proposed adaptive deformation field. (1a): One frame is sufficient in steady state. (1b): When ornaments are attached, additional frames are activated to allow deformations. (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.

The present article extends an earlier conference version [1] by adding new results on deformable mesh registration (6.4), more derivations concerning metrics (4.1), and providing more details based on reviewer comments. The remainder is organized as follows. We summarize the original frame-based simulation method and introduce notations in Section 2. An overview of our adaptive framework is presented in Section 3. We formalize and discuss different criteria for nodal adaptivity in Section 4. The adaptivity of the integration points is then introduced in Section 5. Results obtained with our method are presented and discussed in Section 6, including an application to deformable mesh registration, and we conclude in Section 7 with future work.

1. Related Work

The simulation of viscoelastic solids is a well-studied problem in computer graphics, starting with the early work of Terzopoulos et al. [2]. A survey can be found in [3]. Frame-based models have been proposed [4, 5, 6, 7], and the impressive efficiency of precomputed reduced models has raised a growing interest [8, 9, 10, 11, 12, 13], but run-time adaptivity remains a challenge. The remainder of this review focuses on this issue.

Hutchinson et al. [14] and Ganovelli et al. [15] first combined several resolutions of 2D and 3D solids dynamically deformed by mass-springs. Cotin et al. [16] 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. [17] chose a decomposition scheme that does not generate such nodes. Debunne et al. [18] performed the local explicit integration of non-nested meshes. Grinspun et al. [19] showed that hierarchical shape functions are a generic way to deal with T-nodes. Sifakis et al. [20] 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 [22, 23, 24] or 2D surfaces like cloth [25]. For 3D models, it is an elegant way to deal with cuttings, viscous effects and very thin features [26, 27, 28]. A mesh-less, octree-based adaptive extension of shape matching has been proposed [29]. Besides all these methods based on multiple resolutions, Kim and James [30] 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 analysis 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 applications, where the current configuration and corresponding stress must be used. Wu et al. [17] proposed four criteria based on the curvature of the stress, strain or displacement fields. Debunne et al. [18] considered the Laplacian of the displacement. Lenoir et al. [22] 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 different from the rest state. The problems relative to the criterion thresholds are rarely discussed, even though potential popping artifacts can be problematic: the smaller the thresholds, the smaller the popping artifacts, but also the more difficult to simplify and thus the less efficient.

While our adaptive scheme is primarily targeted at physically-based animation, it can also be interesting to improve the robustness of deformable mesh registration schemes. Finding correspondences between a source (template) mesh and a target mesh or point cloud is a fundamental task in shape acquisition [31] and analysis [32]. As reviewed in [33], local or global correspondence search is generally regularized using a deformation method, that constrains the displacement of the template to a set of feasible transformations. Iterative closest point (ICP) algorithm [34] is the most common procedure to align a source mesh to a target mesh. At each iteration, source point correspondences are locally found by an optimized closest point search [35]. In the original ICP algorithm, the best global linear transformation is found by minimizing distances between source points and their corresponding points. Instead, elastic ICP [36, 37] can be easily performed by treating distance gradients as external forces (*i.e.* springs) applied to a given deformable model. Deformable registration can be more accurate but is however less robust, because a higher number of DOFs makes it more sensitive to local extrema. In contrast, coarse-to-fine registration strategies improve robustness, computational speed, and accuracy. In this paper, we propose to use our adaptive scheme to automatically tune the number of DOFs required during the registration process, and show robustness improvement.

2. 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) whose positions, velocities and forces in world coordinates are stored in state vectors $\mathbf{x}$, $\mathbf{v}$ and $\mathbf{f}$. A node configuration is defined by an affine transformation, represented in homogeneous coordinates by the 4×4 matrix $\mathbf{X}_i$. The world coordinates $\mathbf{x}_i$ of node i are simply the entries in $\mathbf{X}_i$ corresponding to affine transformations. Relative coordinates are obtained similarly from relative transformations: $\mathbf{X}_i^j = \mathbf{X}_j^{-1}\mathbf{X}_i$. A collection of nodes generates a deformation field using a Skeleton Subspace Deformation (SSD) method, also called *skinning* [38]. We use Linear Blend Skinning (LBS), though other methods could be suitable (see *e.g.* [39] for a discussion about SSD techniques). The position of a material point i is defined using a weighted sum of affine displacements:

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

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

$$\mathbf{p} = \mathbf{J}_p\mathbf{x}, \quad \dot{\mathbf{p}} = \mathbf{J}_p\mathbf{v} \tag{2}$$

External forces can be applied directly to the nodes, or to the contact surface of the object. The Principle of Virtual Work implies that nodal forces $\mathbf{f}$ are obtained from skin forces $\mathbf{f_p}$ as $\mathbf{f} = \mathbf{J}_p^T\mathbf{f_p}$. Similarly, the generalized mass matrix for nodes M can be obtained at initialization based on the scalar masses $\mathbf{M}_p$ of skinned particles: $\mathbf{M} = \mathbf{J}_p^T\mathbf{M}_p\mathbf{J}_p$. As shown in [6], differentiating Eq. (1) with respect to material coordinates produces deformation gradients in the current configuration. 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. 5). For each sample, volume 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 [40]. At each time step, we solve a linear equation system

$$\mathbf{A}\Delta\mathbf{v} = \mathbf{b} \tag{3}$$

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).

3. Adaptive Frame-based Simulation

Our first extension to the method presented in Sec. 2 is to attach control nodes to others to reduce the number of independent DOFs. 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 from active nodes using LBS as:

$$\mathbf{X}_i(t) = \sum_{j \in \mathcal{A}} \phi_i^j \mathbf{X}_j(t)\mathbf{X}_i^j(0) \tag{4}$$

where $\mathcal{A}$ 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 rewritten in terms of active nodes only:

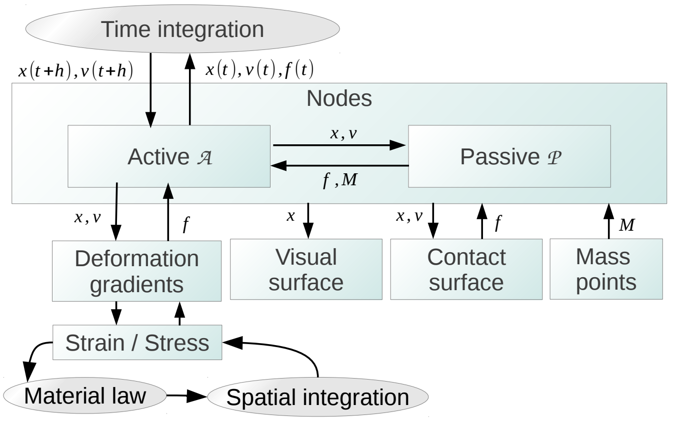


Figure 3: Kinematic structure of the simulation. Our adaptive scheme splits the control nodes into active (*i.e.* independent) nodes and passive (*i.e.* mapped) nodes.

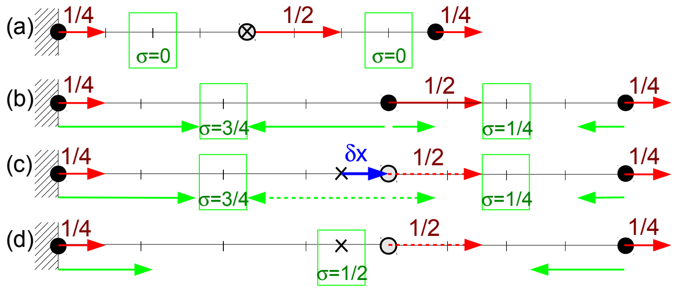


Figure 4: Refinement and simplification. Red and green arrows denote external and internal forces, respectively. Plain circles represent active nodes, while empty circles represent passive nodes attached to their parents, and crosses represent the positions of passive nodes interpolated from their parents 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.

$$\mathbf{p}_i(t) = \sum_{j\in\mathcal{A}} \psi_i^j \mathbf{X}_j(t)\mathbf{X}_j(0)^{-1}\mathbf{p}_i(0) \quad (5)$$

$$\psi_i^j = \phi_i^j + \sum_{k\in\mathcal{P}} \phi_k^j \phi_i^k \quad (6)$$

where $\mathcal{P}$ is the set of passive nodes. These equations generalize similarly to deformation gradients, to obtain a Jacobian matrix $\mathbf{J}_p$ in terms of active nodes alone. This easy composition of LBS is exploited in our node hierarchy (Sec. 4.2.2) and our adaptive spatial integration scheme (Sec. 5). At any time, an active node i can become passive. Since the coefficients used in Eq. (4) 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 between the two configurations $\delta\mathbf{X}_i = \bar{\mathbf{X}}_i^{-1}\mathbf{X}_i$, 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:

$$\mathbf{X}_i(t) = \sum_{j\in\mathcal{A}} \psi_i^j \mathbf{X}_j(t)\mathbf{X}_j(0)^{-1}\mathbf{X}_i(0)\delta\mathbf{X}_i \quad (7)$$

$$\mathbf{u}_i(t) = \mathbf{J}_i\mathbf{v}(t) \quad (8)$$

Our adaptivity criterion is based on comparing the velocity of a passive node attached to nodes of $\mathcal{A}$, 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 $\delta\mathbf{X}$. 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 displacement 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 4 details node adaptivity, while the adaptivity of integration points is presented in Section 5.

4. Adaptive Kinematics

At each time step, our method partitions the nodes into two sets: the *active* nodes, denoted by $\mathcal{A}$, are the currently independent DOFs from which the *passive* nodes, denoted by $\mathcal{P}$, are mapped from the active nodes. We further define a subset $\mathcal{AC} \subset \mathcal{P}$ to be composed of nodes candidate for activation.

Likewise, the deactivation candidate set is a subset $\mathcal{PC} \subset \mathcal{A}$. To decide whether candidate nodes should become passive or active, we compare their velocities in each state (passive and active) and change their status if the velocity difference crosses a certain user-defined threshold η discussed below. At each time step, we compare the velocities in the three following cases:

1. with $\mathcal{A} \setminus \mathcal{PC}$ active and $\mathcal{P} \cup \mathcal{PC}$ passive (coarser resolution)
2. with $\mathcal{A}$ active and $\mathcal{P}$ passive (current resolution)
3. with $\mathcal{A} \cup \mathcal{AC}$ active and $\mathcal{P} \setminus \mathcal{AC}$ 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 $\mathcal{PC}$ as if they were passive, using Eq. (8). For case 3, we simply use an explicit integration for the additional nodes $\mathcal{AC}$, 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 $\mathcal{A}, \mathcal{P}, \mathcal{PC}, \mathcal{AC}$ and finally move on to the next time step.

4.1. Velocity Metrics

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

$$\mathbf{d}_i = \mathbf{J}_i(\mathbf{v} + \Delta\mathbf{v}) - (\mathbf{u}_i + \Delta\mathbf{u}_i) \tag{9}$$

where $\mathbf{J}_i$ is the Jacobian of Eq. (8), 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 then computed as:

$$\mu_i = \|\mathbf{d}_i\|^2_{\mathbf{W}_i} := \frac{1}{2}\mathbf{d}_i^T\mathbf{W}_i\mathbf{d}_i \tag{10}$$

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.

4.1.1. 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, as we now describe. The difference $\mathbf{d} = \mathbf{J}(\mathbf{v} + \Delta\mathbf{v}) - (\mathbf{u} + \Delta\mathbf{u})$ between velocities in the passive and active cases can be seen as a velocity correction due to kinematic constraint forces: $\mathbf{d} = dt\mathbf{M}^{-1}\mathbf{f}_\lambda$ for some force vector $\mathbf{f}_\lambda$, where the corresponding kinematic constraint maintains some frames dependent on others. Therefore, the constraint is holonomic and the associated constraint forces $\mathbf{f}_\lambda$ produce no instantaneous mechanical work:

$$\mathbf{f}_\lambda^T\mathbf{J}(\mathbf{v} + \Delta\mathbf{v}) = 0 \tag{11}$$

This means that $\mathbf{J}(\mathbf{v} + \Delta\mathbf{v})$ and $\mathbf{d}$ are $\mathbf{M}$-orthogonal. It follows that the kinetic energy difference between the active and passive states is simply:

$$\|\mathbf{u} + \Delta\mathbf{u}\|^2_{\mathbf{M}} - \|\mathbf{J}(\mathbf{v} + \Delta\mathbf{v})\|^2_{\mathbf{M}} = \|\mathbf{d}\|^2_{\mathbf{M}} \tag{12}$$

The triangle inequality gives an upper bound on the total change:

$$\|\mathbf{d}\|_{\mathbf{M}} = \Big\| \sum_{i=1}^{k} \hat{\mathbf{d}}_i \Big\|_{\mathbf{M}} \leq \sum_{i=1}^{k} \big\|\hat{\mathbf{d}}_i\big\|_{\mathbf{M}} = \sum_{i=1}^{k} \|\mathbf{d}_i\|_{\mathbf{M}_i} \tag{13}$$

where $\hat{\mathbf{d}}_i = (0, \ldots, \mathbf{d}_i^T, 0, \ldots)^T$ is a column vector whose only non-zero entries are the ones corresponding to DoF i. If we use $\mathbf{W}_i = \mathbf{M}_i$ in Eq. (10), we effectively bound each $\|\mathbf{d}_i\|_{\mathbf{M}_i}$ in Eq. (13), hence the left-hand side $\|\mathbf{d}\|_{\mathbf{M}}$, and finally the total kinetic energy difference. The criterion threshold η can be adapted so that the upper bound in Eq. (13) becomes a small fraction of the current kinetic energy.

4.1.2. Distance to Camera

For computer graphic 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, favoring deactivation. More precisely, if we call $\mathbf{G}_i$ the kinematic mapping between node i and the mesh vertices, obtained by considering mesh vertices as material points in Eq. (1) and Eq. (2), 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:

$$\mathbf{W}_i = \mathbf{G}_i^T\mathbf{Z}\mathbf{G}_i \tag{14}$$

In practice, we use a decreasing exponential for $\mathbf{Z}$ values (1 on the camera near-plane, 0 on the camera far-plane), mimicking the decreasing precision found in the depth buffer during rendering. In our experiments, the exponential decrease resulted in coarser models compared to a linear decrease as the camera distance increased, without noticeable visual quality degradation. 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. Of course, other metrics may be used as well: for instance one may want to penalize distance to a given region of interest.

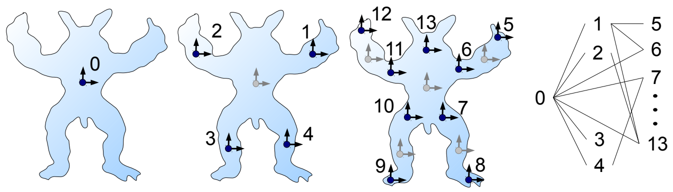


Figure 5: Reference node hierarchy. From left to right: the first three levels, and the dependency graph.

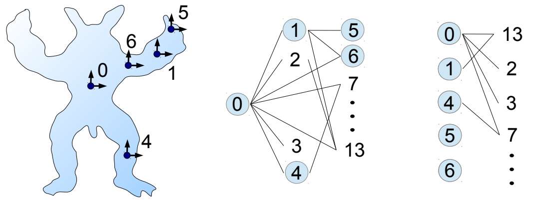


Figure 6: Mechanical hierarchy. Left: active nodes $0, 1, 4, 5, 6$ at a given time. Middle: Reference hierarchy; nodes $2, 3, 7$ are activation candidates; nodes $4, 5, 6$ are deactivation candidates. Right: the resulting two-level contracted graph to be used in the mechanical simulation.

4.2. 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. However, in order to avoid a quadratic number of tests, we pre-compute a node hierarchy and define candidate nodes to be the ones at the interface between passive and active nodes in the hierarchy.

4.2.1. Hierarchy Setup

Our hierarchy is computed at initialization time, as illustrated in Fig. 5. When building each level of the hierarchy, we perform a Lloyd relaxation on a fine voxel grid to spread new control nodes as evenly as possible, taking into account the nodes already created at coarser levels. Shape functions are computed for each level based on the position of inserted nodes and stored in the fine voxel grid as in [7]. Given a node j at a given level l, weights ϕ^i_j relative to its parents i are obtained through interpolation in the grid at level $l-1$. For each non-zero weight, an edge is inserted into the dependency graph, resulting in a generalized hierarchy based on a Directed Acyclic Graph.

4.2.2. Hierarchy Update

The candidates for activation $\mathcal{AC}$ are the passive nodes with all parents active. Conversely, the candidates for deactivation $\mathcal{PC}$ are the active nodes with all children passive, if any, except for the root of the reference hierarchy. In particular, active leaf nodes are always in $\mathcal{PC}$. 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. (5). Similarly, two-level graphs can be obtained for the cases 1 and 3 discussed in the beginning of this section.

5. Adaptive Spatial Integration

We now describe how to adapt the spatial integration of elastic energy in order to further increase computational gains.

5.1. Discretization

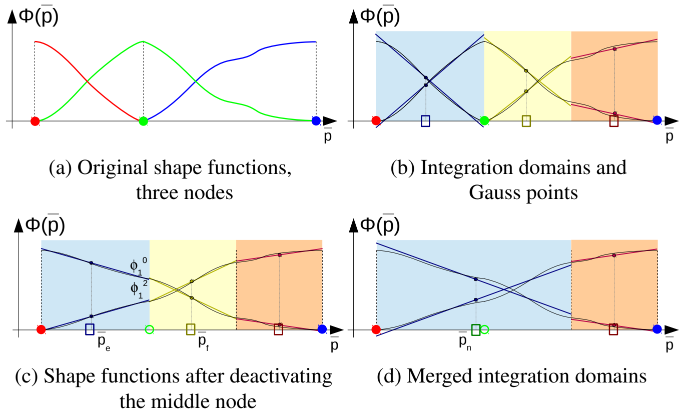
The spatial integration of energy and forces is numerically computed using Gaussian quadrature, a weighted sum of values computed at integration points. Exact quadrature rules are only available for polyhedral domains with polynomial shape functions (e.g. tri-linear hexahedra). In meshless simulation, 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 [4]) 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 $\mathcal{V}e$ centered on point $\bar{\mathbf{p}}_e$, the integral of a function g is given by:

$$\int_{\bar{\mathbf{p}}\in\mathcal{V}e} g \approx \mathbf{g}^T \int_{\bar{\mathbf{p}}\in\mathcal{V}e} (\bar{\mathbf{p}} - \bar{\mathbf{p}}_e)^{(n)} = \mathbf{g}^T \bar{\mathbf{g}}_e \tag{15}$$

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 integrated across $\mathcal{V}e$ which can be computed at initialization time by looping over the voxels of an arbitrarily fine rasterization. The approximation of Eq. (15) 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 $\mathcal{V}e$. The linearity error is given by:



(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.

$$\varepsilon(\mathbf{c}) = \int_{\mathcal{V}e} (\phi_i(\bar{\mathbf{p}}) - \mathbf{c}^T \bar{\mathbf{p}}^{(1)})^2 \quad (16)$$

$$= \mathbf{c}^T A^e \mathbf{c} - 2\mathbf{c}^T B_i^e + C_i^e \quad (17)$$

$$\text{with: } A^e = \int_{\mathcal{V}e} \bar{\mathbf{p}}^{(1)} \bar{\mathbf{p}}^{(1)T} \quad (18)$$

$$B_i^e = \int_{\mathcal{V}e} \phi_i(\bar{\mathbf{p}}) \bar{\mathbf{p}}^{(1)} \quad (19)$$

$$C_i^e = \int_{\mathcal{V}e} \phi_i(\bar{\mathbf{p}})^2 \quad (20)$$

We solve for the 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.

5.2. *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. (6). This allows us to merge integration points sharing the same set of active nodes (in $\mathcal{A} \cup \mathcal{AC}$), 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. (15) 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. (5) 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.

6. Results

We now report experimental results obtained with our method, demonstrating its interest for computer graphics. We also propose an application of our technique to the deformable mesh registration problem.

6.1. *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 fully coarsened 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 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.

6.2. *Complex Scenes*

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

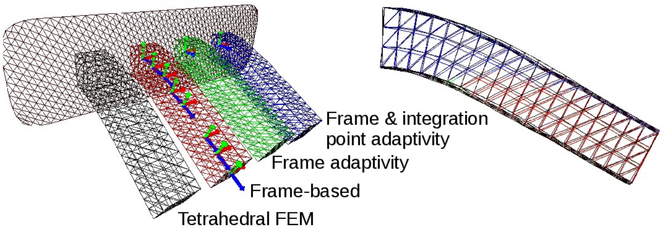
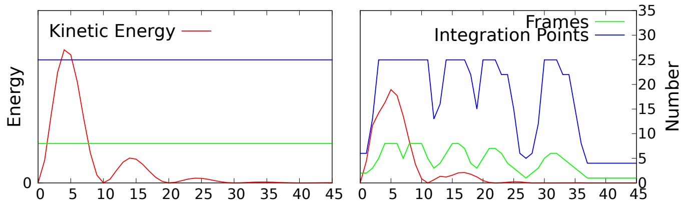


Figure 8: Four cantilever beams at equilibrium with the same properties and loading (fixed on one side, and subject to gravity). Right: perspective side view of the same configuration, showing that our adaptivity framework produces results similar (up to the depth-buffer precision, hence the color changes) to the non-adaptive frame-based approach.



(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).

6.2.1. *Christmas Tree*

A Christmas tree (Fig. 1) with a stiff trunk and more flexible branches, with 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.

6.2.2. *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.

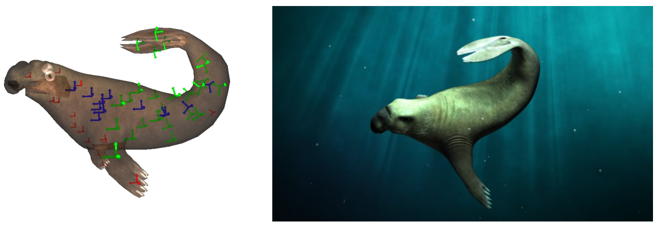


Figure 10: 40 adaptive, elastic frames *(green: active, red: passive)* adding secondary motion on a (deliberately short) kinematic skeleton corresponding to 12 (blue) frames.

6.2.3. *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 demonstrates that our method allows simplifications in non-equilibrium states.

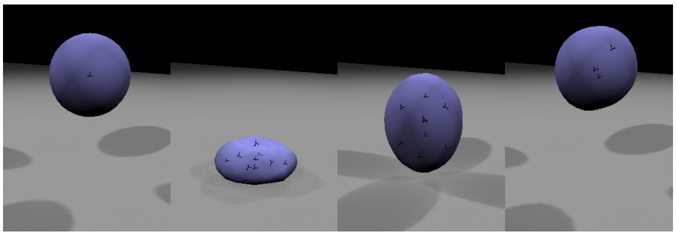


Figure 11: A falling deformable ball with unilateral contacts.

6.2.4. *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 as the shoe goes away. In this example of multi-body adaptivity, the root node configuration has little importance and simply corresponds to a global affine transformation of all the mechanical bodies. The second level of the hierarchy consists in one node per body, and the remaining levels are restricted to each object (*i.e.* no cross-object node influence, though our method allows this).

(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.

6.2.5. *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. (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.

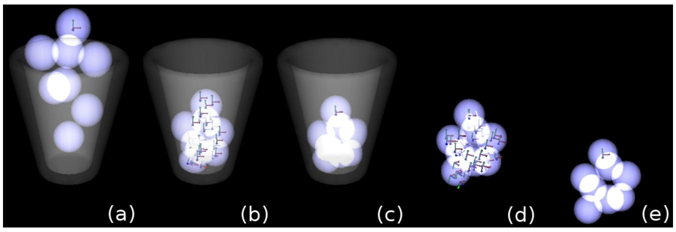


Figure 13: Eight deformable balls stacking up in a glass, which is eventually removed.

6.2.6. *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.

6.3. *Performance*

In the various scenarios described above, our technique allows a significant reduction of both kinematic DOFs and integration points, as presented in Table 2. 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 1 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 1: Speedups for a salad of one and 18 armadillos at various maximal resolutions (including collision timing)

6.4. *Application to Mesh Registration*

Using our adaptive scheme, the number of DOFs can be progressively increased as needed during the registration process, as shown in Fig. 14. Our approach is the following: we start with a fully coarsened model and apply the deformable registration procedure. Springs between the deformable surface and corresponding points on the target surface are created at each iteration. For a given spring stiffness, nodes become active as needed to produce the required, increasingly local deformations, minimizing both external and internal energy. After equilibrium is reached, our adaptive kinematic model comes back to its fully coarsened state. When this happens, we progressively increase the registration spring stiffness, then again let the adaptive model deform accordingly. We iterate this process until the distance to the target surface becomes smaller than a user-defined threshold. By using our adaptive kinematics, the deformation field seamlessly transitions from global to local transformations as needed, implementing the coarse-to-fine registration strategy. In addition, parts of the mesh that are already registered will not be refined again as other parts continue to undergo deformations, if these parts belong to separate branches of the node hierarchy. Consider for instance the node hierarchy presented in Fig. 5: if the legs are already registered (nodes $3, 4, 7, 8, 9, 10$), the arm registration (nodes $1, 2, 5, 6, 11, 12$) will not trigger leg node activation since these nodes belong to separate subgraphs, thus avoiding unnecessary computations. A comparison between adaptive and non-adaptive registration is presented in Fig. 15: our adaptive kinematics produce better results even though the registration potential forces are the same in both cases. In this example, the overall computational time was about 3 minutes. Since most of it was spent on closest point search, we did not notice significant computational gains between the adaptive and non-adaptive cases.

7. Conclusion and Perspectives

We introduced a novel method for the run-time adaptivity of elastic models. Our method requires few pre-processing (a 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

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 2: Adaptivity performances and timings.

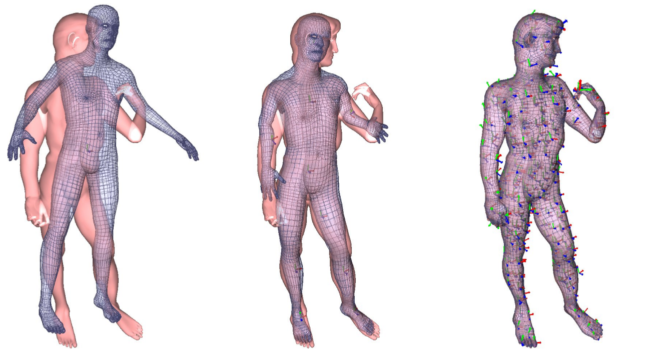


Figure 14: Deformable mesh registration of an adaptive deformable model *(blue)* to a target mesh *(red)*. Left to right: while initially fully coarsened, the deformable model is progressively refined, automatically producing increasingly local deformations.

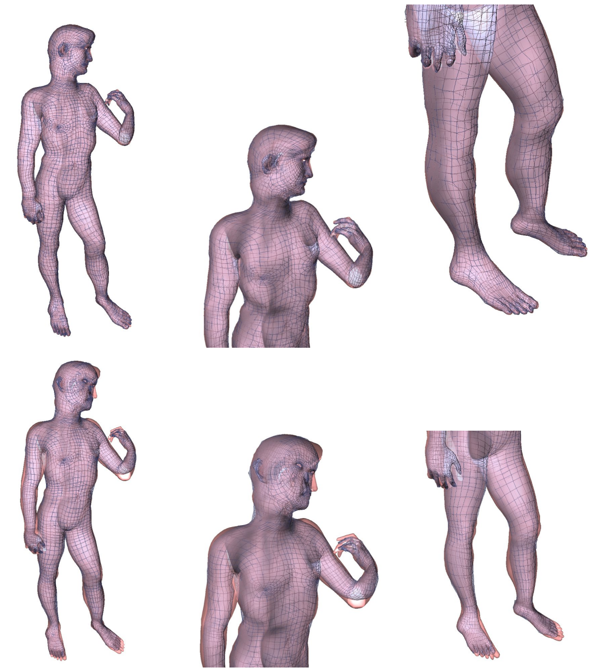


Figure 15: Comparison of deformable registration results for adaptive *(top)* and non-adaptive *(bottom)* kinematics. Our coarse-to-fine scheme produces a tighter fit, given the same registration potential forces.

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 kinematic 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. Finally, we presented an application of this scheme to the deformable mesh registration problem, for which a coarse-to-fine deformation strategy can be very easily implemented.

In future work, we will address the question of stiffness discontinuities and the design of scenario-dependent frame hierarchies. We will also perform a more in-depth analysis and evaluation of our adaptive scheme in the context of mesh registration. Finally, the presented technique is likely to be generalizable to non-frame kinematic DOFs, which could provide a basis for a fully generic adaptivity framework.

References

[1] Tournier M, Nesme M, Faure F, Gilles B. Seamless adaptivity of elastic models. In: Proceedings of the 2014 Graphics Interface Conference. Canadian Information Processing Society; 2014, p. 17–24.

[2] Terzopoulos D, Platt J, Barr A, Fleischer K. Elastically deformable models. In: ACM Siggraph Computer Graphics; vol. 21. ACM; 1987, p. 205–14.

[3] Nealen A, Müller M, Keiser R, Boxerman E, Carlson M. Physically based deformable models in computer graphics. In: Computer Graphics Forum; vol. 25. Wiley Online Library; 2006, p. 809–36.

[4] Martin S, Kaufmann P, Botsch M, Grinspun E, Gross M. Unified simulation of elastic rods, shells, and solids; vol. 29. ACM; 2010.

[5] Müller M, Chentanez N. Solid simulation with oriented particles. In: ACM Transactions on Graphics (TOG); vol. 30. ACM; 2011, p. 92–.

[6] Gilles B, Bousquet G, Faure F, Pai DK. Frame-based elastic models. In: ACM Transactions on Graphics (TOG); vol. 30. ACM; 2011, p. 15–.

[7] Faure F, Gilles B, Bousquet G, Pai DK. Sparse meshless models of complex deformable solids. In: ACM Transactions on Graphics (TOG); vol. 30. ACM; 2011, p. 73–.

[8] Kry PG, James DL, Pai DK. Eigenskin: real time large deformation character skinning in hardware. In: Proc. ACM SIGGRAPH/Eurographics Symposium on Computer animation. ISBN 1-58113-573-4; 2002, p. 153–9. doi:http://doi.acm.org/10.1145/545261.545286.

[9] Barbič J, James DL. Real-time subspace integration for St. Venant-Kirchhoff deformable models. ACM Transactions on Graphics (Proc SIGGRAPH) 2005;24(3):982–90.

[10] Barbič J, Zhao Y. Real-time large-deformation substructuring. In: ACM Transactions on Graphics (TOG); vol. 30. ACM; 2011, p. 91–.

[11] Kim J, Pollard NS. Fast simulation of skeleton-driven deformable body characters. ACM Transactions on Graphics 2011;30.

[12] Hahn F, Martin S, Thomaszewski B, Sumner R, Coros S, Gross M. Rig-space physics. In: ACM Transactions on Graphics (TOG); vol. 31. ACM; 2012, p. 72–.

[13] Hildebrandt K, Schulz C, von Tycowicz C, Polthier K. Interactive space-time control of deformable objects. In: ACM Transactions on Graphics (TOG); vol. 31. ACM; 2012, p. 71–.

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

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

[16] Delingette H, Cotin S, Ayache N. A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training and simulation. In: Computer animation, 1999. Proceedings. IEEE; 1999, p. 70–81.

[17] Wu X, Downes MS, Goktekin T, Tendick F. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. In: Computer Graphics Forum; vol. 20. Wiley Online Library; 2001, p. 349–58.

[18] Debunne G, Desbrun M, Cani MP, Barr AH. Dynamic real-time deformations using space & time adaptive sampling. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques. ACM; 2001, p. 31–6.

[19] Grinspun E, Krysl P, Schröder P. Charms: a simple framework for adaptive simulation. In: ACM Transactions on Graphics (TOG); vol. 21. ACM; 2002, p. 281–90.

[20] Sifakis E, Shinar T, Irving G, Fedkiw R. Hybrid simulation of deformable solids. In: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation. Eurographics Association; 2007, p. 81–90.

[21] Martin S, Kaufmann P, Botsch M, Wicke M, Gross M. Polyhedral finite elements using harmonic basis functions. In: Computer Graphics Forum; vol. 27. Wiley Online Library; 2008, p. 1521–9.

[22] Lenoir J, Grisoni L, Chaillou C, Meseure P. Adaptive resolution of 1d mechanical b-spline. In: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia. ACM; 2005, p. 395–403.

[23] Spillmann J, Teschner M. An adaptive contact model for the robust simulation of knots. In: Computer Graphics Forum; vol. 27. Wiley Online Library; 2008, p. 497–506.

[24] Servin M, Lacoursiere C, Nordfelth F, Bodin K. Hybrid, multiresolution wires with massless frictional contacts. In: Visualization and Computer Graphics, IEEE Transactions on; vol. 17. IEEE; 2011, p. 970–82.

[25] Narain R, Samii A, O'Brien JF. Adaptive anisotropic remeshing for cloth simulation. In: ACM Transactions on Graphics (TOG); vol. 31. ACM; 2012, p. 152–.

[26] Bargteil AW, Wojtan C, Hodgins JK, Turk G. A finite element method for animating large viscoplastic flow. In: ACM Transactions on Graphics (TOG); vol. 26. ACM; 2007, p. 16–.

[27] Wojtan C, Turk G. Fast viscoelastic behavior with thin features. In: ACM Transactions on Graphics (TOG); vol. 27. ACM; 2008, p. 47–.

[28] Wicke M, Ritchie D, Klingner BM, Burke S, Shewchuk JR, O'Brien JF. Dynamic local remeshing for elastoplastic simulation. In: ACM Transactions on graphics (TOG); vol. 29. ACM; 2010, p. 49–.

[29] Steinemann D, Otaduy MA, Gross M. Fast adaptive shape matching deformations. In: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. Eurographics Association; 2008, p. 87–94.

[30] Kim T, James DL. Skipping steps in deformable simulation with online model reduction. In: ACM Transactions on Graphics (TOG); vol. 28. ACM; 2009, p. 123–.

[31] Li H, Sumner R, Pauly M. Global correspondence optimization for nonrigid registration of depth scans. Computer Graphics Forum 2008;27(5):1421–30.

[32] Allen B, Curless B, Popović Z. The space of human body shapes: Reconstruction and parameterization from range scans. ACM Trans Graph 2003;22(3):587–94.

[33] van Kaick O, Zhang H, Hamarneh G, Cohen-Or D. A survey on shape correspondence. Computer Graphics Forum 2011;30(6):1681–707.

[34] Besl P, McKay N. A method for registration of 3-d shapes. IEEE Trans on Pattern Analysis and Machine Intelligence 1992;14(2):239–56.

[35] Rusinkiewicz S, Levoy M. Efficient variants of the icp algorithm. 3-D Digital Imaging and Modeling 2001;:145–52.

[36] Amberg B, Romdhani S, Vetter T. Optimal step nonrigid icp algorithms for surface registration. In: CVPR'07. 2007,.

[37] Gilles B, Reveret L, Pai D. Creating and animating subject-specific anatomical models. Computer Graphics Forum 2010;29(8):2340–51.

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

[39] Kavan L, Collins S, Žára J, O'Sullivan C. Skinning with dual quaternions. In: Proceedings of the 2007 symposium on Interactive 3D graphics and games. ACM; 2007, p. 39–46.

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