



**HAL**  
open science

## Velocity-based Adaptivity of Deformable Models

Maxime Tournier, Matthieu Nesme, François Faure, Benjamin Gilles

► **To cite this version:**

Maxime Tournier, Matthieu Nesme, François Faure, Benjamin Gilles. Velocity-based Adaptivity of Deformable Models. *Computers and Graphics*, 2014, 45, pp.75-85. 10.1016/j.cag.2014.08.004 . hal-01076338

**HAL Id: hal-01076338**

**<https://inria.hal.science/hal-01076338>**

Submitted on 5 Dec 2014

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Velocity-based Adaptivity of Deformable Models

Maxime Tournier<sup>1</sup>, Matthieu Nesme<sup>2</sup>, Francois Faure<sup>2</sup>, Benjamin Gilles<sup>1</sup>

## 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

1 The stunning quality of high-resolution physically-based an-  
2 imations of deformable solids requires complex deformable mod-  
3 els with large numbers of independent Degrees Of Freedom  
4 (DOF) which result in large equation systems for solving dy-  
5 namics, and high computation times. On the other hand, the  
6 thrilling user experience provided by interactive simulations can  
7 only be achieved using fast computation times which preclude  
8 the use of high-resolution models. Reconciling these two con-  
9 tradictory goals requires adaptive models to efficiently manage  
10 the number of DOFs, by refining the model where necessary  
11 and by coarsening it where possible. Mesh-based deformations  
12 can be seamlessly refined by subdividing elements and interpo-  
13 lating new nodes within these. However, seamless coarsening  
14 can be performed only when the fine nodes are back to their  
15 original position with respect to their higher-level elements,  
16 which only happens in the locally undeformed configurations  
17 (*i.e.* with null strain). Otherwise, a popping artifact (*i.e.* an in-  
18 stantaneous change of shape) occurs, which not only violates  
19 the laws of physics, but it is also visually disturbing for the user.  
20 Simplifying objects in deformed configurations, as presented in  
21 Fig. 1c, has thus not been possible with previous adaptive ap-  
22 proaches, unless the elements are small or far enough from the  
23 user. This may explain why extreme coarsening has rarely been  
24 proposed, and adaptive FEM models typically range from mod-  
25 erate to high complexity.

26 We introduce a new approach of adaptivity to mechanically  
27 simplify objects in arbitrarily deformed configurations, while  
28 exactly maintaining their current shape and controlling the ve-  
29 locity discontinuity, which we call seamless adaptivity. It ex-  
30 tends a frame-based meshless method and naturally exploits the  
31 ability to attach frames to others in arbitrary relative positions,  
32 as illustrated in Fig. 2. In this example, a straight beam is ini-

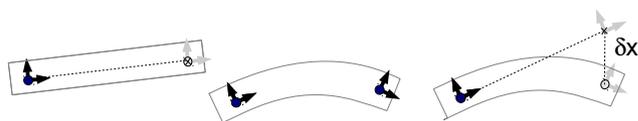


Figure 2: Seamless coarsening in a deformed state. Left: reference shape, one active frame in black, and a passive frame in grey attached using a relative transformation (dotted line). Middle: activating the frame let it to move freely and deform the object. Right: deactivated frame in a deformed configuration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

33 tially animated using a single moving frame, while another con-  
34 trol frame is attached to it. We then detach the child frame to  
35 allow the beam to bend as needed. If the beam deformation  
36 reaches a steady state, the *velocity* field can again be obtained  
37 from the moving frame alone, and the *shape* can be frozen in  
38 the deformed state by applying an offset to the child frame  
39 reference position relative to the moving frame. Setting the offset  
40 to the current relative position removes mechanical DOFs with-  
41 out altering the current shape of the object. This deformation is  
42 reversible. If the external loading applied to the object changes,  
43 we can mechanically refine the model again (*i.e.* activate the  
44 passive frame) to allow the object to recover its initial shape or  
45 to undergo new deformations. The ability to dynamically adapt  
46 the deformation field even in non-rest configuration is the spe-  
47 cific feature of our approach, which dramatically enhances the  
48 opportunities for coarsening mechanical models compared with  
49 previous methods.

50 Our specific contributions are (1) a deformation method based  
51 on a generalized frame hierarchy for dynamically tuning the  
52 complexity of deformable solids with seamless transitions; (2)  
53 a novel simplification and refinement criterion based on veloc-  
54 ity, which allows us to simplify the deformation model in de-  
55 formed configurations, and (3) a method to dynamically adapt  
56 the integration points and enforce the continuity of forces across  
57 changes of resolution.

<sup>1</sup>INRIA, LIRMM-CNRS, Université de Montpellier 2

<sup>2</sup>INRIA, LJK-CNRS, Université de Grenoble



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.

58 The present article extends an earlier conference version [1]  
 59 by adding new results on deformable mesh registration (6.4),  
 60 more derivations concerning metrics (4.1), and providing more  
 61 details based on reviewer comments. The remainder is orga-  
 62 nized as follows. We summarize the original frame-based simu-  
 63 lation method and introduce notations in Section 2. An overview  
 64 of our adaptive framework is presented in Section 3. We formal-  
 65 ize and discuss different criteria for nodal adaptivity in Section  
 66 4. The adaptivity of the integration points is then intro-  
 67 duced in Section 5. Results obtained with our method are pre-  
 68 sented and discussed in Section 6, including an application to  
 69 deformable mesh registration, and we conclude in Section 7  
 70 with future work.

## 71 1. Related Work

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

79 Hutchinson et al. [14] and Ganovelli et al. [15] first com-  
 80 bined several resolutions of 2D and 3D solids dynamically de-  
 81 formed by mass-springs. Cotin et al. [16] combined two me-  
 82 chanical models to simulate various parts of the same object.  
 83 Most adaptive methods are based on meshes at multiple resolu-  
 84 tions. Mixing different mesh sizes can result in T-nodes that are  
 85 mechanically complex to manage in the Finite Element Method  
 86 (FEM). Wu et al. [17] chose a decomposition scheme that does  
 87 not generate such nodes. Debonne et al. [18] performed the  
 88 local explicit integration of non-nested meshes. Grinspun et  
 89 al. [19] showed that hierarchical shape functions are a generic  
 90 way to deal with T-nodes. Sifakis et al. [20] constrained T-  
 91 nodes within other independent nodes. Martin et al. [21] solved  
 92 multi-resolution junctions with polyhedral elements. Several

93 authors proposed to generate on the fly a valid mesh with dense  
 94 and fine zones. Real-time remeshing is feasible for 1D ele-  
 95 ments such as rods and wires [22, 23, 24] or 2D surfaces like  
 96 cloth [25]. For 3D models, it is an elegant way to deal with  
 97 cuttings, viscous effects and very thin features [26, 27, 28]. A  
 98 mesh-less, octree-based adaptive extension of shape matching  
 99 has been proposed [29]. Besides all these methods based on  
 100 multiple resolutions, Kim and James [30] take a more alge-  
 101 braic approach, where the displacement field is decomposed  
 102 on a small, dynamically updated, basis of orthogonal vectors,  
 103 while a small set of carefully chosen integration points are used  
 104 to compute the forces. In contrast to these works, our method  
 105 relies on velocity field analysis and a meshless discretization.

106 Numerous error estimators for refinement have been pro-  
 107 posed in conventional FEM analysis. For static analysis, they  
 108 are generally based on a precomputed stress field. This is not  
 109 feasible in real-time applications, where the current configura-  
 110 tion and corresponding stress must be used. Wu et al. [17] pro-  
 111 posed four criteria based on the curvature of the stress, strain or  
 112 displacement fields. Debonne et al. [18] considered the Lapla-  
 113 cian of the displacement. Lenoir et al. [22] refined parts in  
 114 contact for wire simulation. These approaches refine the ob-  
 115 jects where they are the most deformed, and they are not able to  
 116 save computation time in equilibrium states different from the  
 117 rest state. The problems relative to the criterion thresholds are  
 118 rarely discussed, even though potential popping artifacts can be  
 119 problematic: the smaller the thresholds, the smaller the popping  
 120 artifacts, but also the more difficult to simplify and thus the less  
 121 efficient.

122 While our adaptive scheme is primarily targeted at physically-  
 123 based animation, it can also be interesting to improve the ro-  
 124 bustness of deformable mesh registration schemes. Finding  
 125 correspondences between a source (template) mesh and a tar-  
 126 get mesh or point cloud is a fundamental task in shape acqui-  
 127 sition [31] and analysis [32]. As reviewed in [33], local or  
 128 global correspondence search is generally regularized using a  
 129 deformation method, that constrains the displacement of the

130 template to a set of feasible transformations. Iterative closest  
 131 point (ICP) algorithm [34] is the most common procedure to  
 132 align a source mesh to a target mesh. At each iteration, source  
 133 point correspondences are locally found by an optimized closest  
 134 point search [35]. In the original ICP algorithm, the best  
 135 global linear transformation is found by minimizing distances  
 136 between source points and their corresponding points. Instead,  
 137 elastic ICP [36, 37] can be easily performed by treating distance  
 138 gradients as external forces (*i.e.* springs) applied to a given de-  
 139 formable model. Deformable registration can be more accurate  
 140 but is however less robust, because a higher number of DOFs  
 141 makes it more sensitive to local extrema. In contrast, coarse-to-  
 142 fine registration strategies improve robustness, computational  
 143 speed, and accuracy. In this paper, we propose to use our adap-  
 144 tive scheme to automatically tune the number of DOFs required  
 145 during the registration process, and show robustness improve-  
 146 ment.

## 147 2. Frame-based Simulation Method

148 In this section we summarize the method that our contribu-  
 149 tion extends, and we introduce notations and basic equations.  
 150 The method of [7] performs the physical simulation of vis-  
 151 coelastic solids using a hyperelastic formulation. The control  
 152 nodes are moving frames with 12 Degrees of Freedom (DOF)  
 153 whose positions, velocities and forces in world coordinates are  
 154 stored in state vectors  $\mathbf{x}$ ,  $\mathbf{v}$  and  $\mathbf{f}$ . A node configuration is de-  
 155 fined by an affine transformation, represented in homogeneous  
 156 coordinates by the  $4 \times 4$  matrix  $\mathbf{X}_i$ . The world coordinates  $\mathbf{x}_i$  of  
 157 node  $i$  are simply the entries in  $\mathbf{X}_i$  corresponding to affine trans-  
 158 formations. Relative coordinates are obtained similarly from  
 159 relative transformations:  $\mathbf{X}_i^j = \mathbf{X}_j^{-1}\mathbf{X}_i$ . A collection of nodes  
 160 generates a deformation field using a Skeleton Subspace Deform-  
 161 ation (SSD) method, also called *skinning* [38]. We use Linear  
 162 Blend Skinning (LBS), though other methods could be suitable  
 163 (see *e.g.* [39] for a discussion about SSD techniques). The po-  
 164 sition of a material point  $i$  is defined using a weighted sum of  
 165 affine displacements:

$$166 \quad \mathbf{p}_i(t) = \sum_{j \in \mathcal{N}} \phi_j^j \mathbf{X}_j(t) \mathbf{X}_j(0)^{-1} \mathbf{p}_i(0) \quad (1)$$

167 where  $\mathcal{N}$  is the set of all control nodes, and  $\phi_j^j$  is the value  
 168 of the shape function of node  $j$  at material position  $\mathbf{p}_i(0)$ , com-  
 169 puted at initialization time using distance ratios as in [7]. Spa-  
 170 tially varying shape functions allow more complex deforma-  
 171 tions. Similar to nodes, the state of all skinned points are stored  
 172 as vectors:  $\mathbf{p}$ ,  $\dot{\mathbf{p}}$ , and  $\mathbf{f}_p$ . Eq. (1) is linear in node coordinates,  
 173 therefore a *constant* Jacobian matrix  $\mathbf{J}_p$  can be assembled at  
 174 initialization, relating node coordinates to skinned point coor-  
 175 dinates:

$$176 \quad \mathbf{p} = \mathbf{J}_p \mathbf{x}, \quad \dot{\mathbf{p}} = \mathbf{J}_p \mathbf{v} \quad (2)$$

177 External forces can be applied directly to the nodes, or to  
 178 the contact surface of the object. The Principle of Virtual Work  
 179 implies that nodal forces  $\mathbf{f}$  are obtained from skin forces  $\mathbf{f}_p$  as

178  $\mathbf{f} = \mathbf{J}_p^T \mathbf{f}_p$ . Similarly, the generalized mass matrix for nodes  $M$   
 179 can be obtained at initialization based on the scalar masses  $\mathbf{M}_p$   
 180 of skinned particles:  $\mathbf{M} = \mathbf{J}_p^T \mathbf{M}_p \mathbf{J}_p$ . As shown in [6], differ-  
 181 entiating Eq. (1) with respect to material coordinates produces  
 182 deformation gradients in the current configuration. By map-  
 183 ping deformation gradients to strains (such as Cauchy, Green-  
 184 Lagrange or corotational), and applying a constitutive law (such  
 185 as Hooke or Mooney-Rivlin), we can compute the elastic poten-  
 186 tial energy density at any location. After spatial integration and  
 187 differentiation with respect to the degrees of freedom, forces  
 188 can be computed and propagated back to the nodes.

189 We use different discretizations for visual surfaces, con-  
 190 tact surfaces, mass and elasticity (potential energy integration  
 191 points). Masses are precomputed using a dense volumetric ras-  
 192 terization, where voxels are seen as point masses. Deforma-  
 193 tion gradient samples (*i.e.* Gauss points) are distributed so as to  
 194 minimize the numerical integration error (see Sec. 5). For each  
 195 sample, volume moments are precomputed from the fine voxel  
 196 grid and associated with local material properties.

197 The method is agnostic with respect to the way we solve  
 198 the equations of motion. We apply an implicit time integra-  
 199 tion to maintain stability in case of high stiffness or large time  
 200 steps [40]. At each time step, we solve a linear equation system

$$201 \quad \mathbf{A} \Delta \mathbf{v} = \mathbf{b} \quad (3)$$

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

## 209 3. Adaptive Frame-based Simulation

210 Our first extension to the method presented in Sec. 2 is to  
 211 attach control nodes to others to reduce the number of inde-  
 212 pendent DOFs. This amounts to adding an extra block to the  
 213 kinematic structure of the model, as shown in Fig. 3.

214 The independent state vectors are restricted to the active  
 215 nodes. At each time step, the dynamics equation is solved to  
 216 update the positions and velocities of the active nodes, then the  
 217 changes are propagated to the passive nodes, then to the skin  
 218 points and the material integration points. The forces are prop-  
 219 agated the other way round. When a node  $i$  is passive, its matrix  
 220 is computed from active nodes using LBS as:

$$221 \quad \mathbf{X}_i(t) = \sum_{j \in \mathcal{A}} \phi_i^j \mathbf{X}_j(t) \mathbf{X}_i^j(0) \quad (4)$$

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

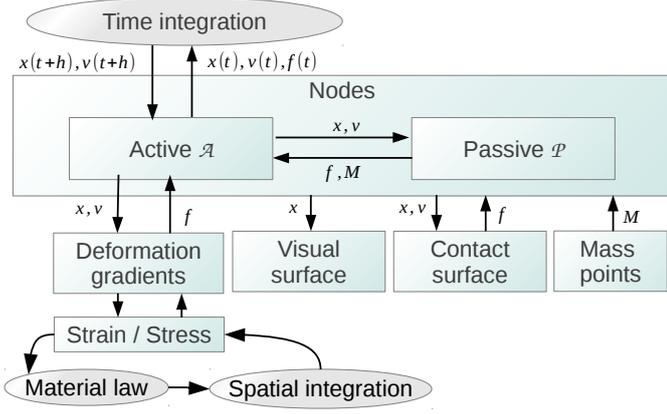


Figure 3: Kinematic structure of the simulation. Our adaptive scheme splits the control nodes into active (*i.e.* independent) nodes and passive (*i.e.* mapped) nodes.

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

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

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

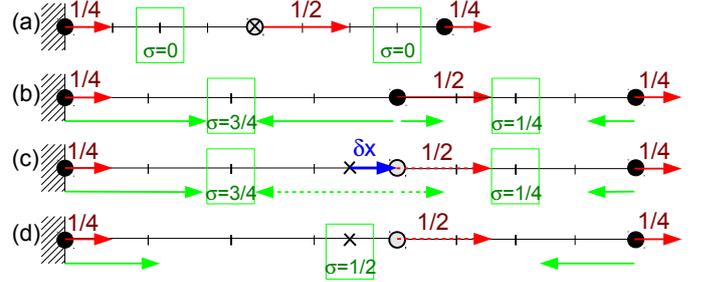


Figure 4: Refinement and simplification. Red and green arrows denote external and internal forces, respectively. Plain circles represent active nodes, while empty circles represent passive nodes attached to their parents, and crosses represent the positions of passive nodes interpolated from their parents positions. Dashed lines are used to denote forces divided up among the parent nodes. Rectangles denote integration points, where the stresses  $\sigma$  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.

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.

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

- 291 1. with  $\mathcal{A} \setminus \mathcal{PC}$  active and  $\mathcal{P} \cup \mathcal{PC}$  passive (coarser resolu-  
 292 tion)
- 293 2. with  $\mathcal{A}$  active and  $\mathcal{P}$  passive (current resolution)
- 294 3. with  $\mathcal{A} \cup \mathcal{AC}$  active and  $\mathcal{P} \setminus \mathcal{AC}$  passive (finer resolution)

295  
 296 We avoid solving the three implicit integrations, noticing  
 297 that cases 1 and 3 are only used to compute the adaptivity cri-  
 298 terion. Instead of performing the implicit integration for case  
 299 1, we use the solution given by 2 and we compute the veloci-  
 300 ties of the frames in  $\mathcal{PC}$  as if they were passive, using Eq. (8).  
 301 For case 3, we simply use an explicit integration for the addi-  
 302 tional nodes  $\mathcal{AC}$ , in linear time using a lumped mass matrix. In  
 303 practice, we only noticed small differences with a fully implicit  
 304 integration. At worse, overshooting due to explicit integration  
 305 temporarily activates too many nodes.

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

#### 310 4.1. Velocity Metrics

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

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

313 where  $\mathbf{J}_i$  is the Jacobian of Eq. (8), and  $\Delta\mathbf{v}, \Delta\mathbf{u}_i$  are the  
 314 velocity updates computed by time integration, respectively in  
 315 the case where the candidate node is passive and active. Note  
 316 that for the activation criterion computed using explicit integra-  
 317 tion (case 3), this reduces to the generalized velocity difference  
 318  $\mathbf{d}_i = \mathbf{J}_i\Delta\mathbf{v} - dt\tilde{\mathbf{M}}_i^{-1}\mathbf{f}_i$  where  $\tilde{\mathbf{M}}_i$  is the lumped mass matrix block  
 319 of node  $i$ ,  $\mathbf{f}_i$  its net external force and  $dt$  is the time step, which  
 320 is a difference in *acceleration* up to  $dt$ . A measure of  $\mathbf{d}_i$  is then  
 321 computed as:

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

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

##### 327 4.1.1. Kinetic Energy

328 As the nodes are transitioning between passive and active  
 329 states, a velocity discontinuity may occur. In order to prevent  
 330 instabilities, a natural approach is to bound the associated ki-  
 331 netic energy discontinuity, as we now describe. The difference  
 332  $\mathbf{d} = \mathbf{J}(\mathbf{v} + \Delta\mathbf{v}) - (\mathbf{u} + \Delta\mathbf{u})$  between velocities in the passive and

333 active cases can be seen as a velocity correction due to kine-  
 334 matic constraint forces:  $\mathbf{d} = dt\mathbf{M}^{-1}\mathbf{f}_\lambda$  for some force vector  $\mathbf{f}_\lambda$ ,  
 335 where the corresponding kinematic constraint maintains some  
 336 frames dependent on others. Therefore, the constraint is holo-  
 337 nomic and the associated constraint forces  $\mathbf{f}_\lambda$  produce no instan-  
 338 taneous mechanical work:

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

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

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

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

$$\|\mathbf{d}\|_{\mathbf{M}} = \left\| \sum_{i=1}^k \hat{\mathbf{d}}_i \right\|_{\mathbf{M}} \leq \sum_{i=1}^k \|\hat{\mathbf{d}}_i\|_{\mathbf{M}} = \sum_{i=1}^k \|\mathbf{d}_i\|_{\mathbf{M}_i} \quad (13)$$

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

##### 351 4.1.2. Distance to Camera

352 For computer graphic applications, one is usually ready to  
 353 sacrifice precision for speed as long as the approximation is  
 354 not visible to the user. To this end, we can measure velocity  
 355 differences according to the distance to the camera of the asso-  
 356 ciated visual mesh, so that motion happening far from the cam-  
 357 era will produce lower measures, favoring deactivation. More  
 358 precisely, if we call  $\mathbf{G}_i$  the kinematic mapping between node  $i$   
 359 and the mesh vertices, obtained by considering mesh vertices as  
 360 material points in Eq. (1) and Eq. (2), and  $\mathbf{Z}$  a diagonal matrix  
 361 with positive values decreasing along with the distance between  
 362 mesh vertices and the camera, the criterion metric is then given  
 363 by:

$$\mathbf{W}_i = \mathbf{G}_i^T \mathbf{Z} \mathbf{G}_i \quad (14)$$

364 In practice, we use a decreasing exponential for  $\mathbf{Z}$  values  
 365 (1 on the camera near-plane, 0 on the camera far-plane), mim-  
 366 icking the decreasing precision found in the depth buffer dur-  
 367 ing rendering. In our experiments, the exponential decrease  
 368 resulted in coarser models compared to a linear decrease as  
 369 the camera distance increased, without noticeable visual qual-  
 370 ity degradation. The two metrics can also be combined by re-  
 371 taining the minimum of their values: simplification is then fa-  
 372 vored far from the camera, where the distance metric is always  
 373 small, while the kinetic energy metric is used close to the cam-  
 374 era, where the distance metric is always large. Of course, other  
 375 metrics may be used as well: for instance one may want to pe-  
 376 nalize distance to a given region of interest.

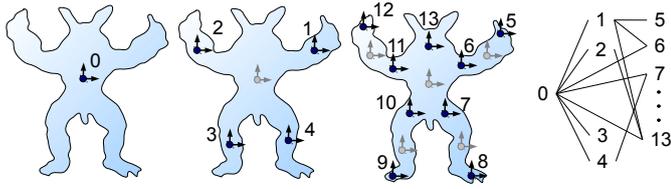


Figure 5: Reference node hierarchy. From left to right: the first three levels, and the dependency graph.

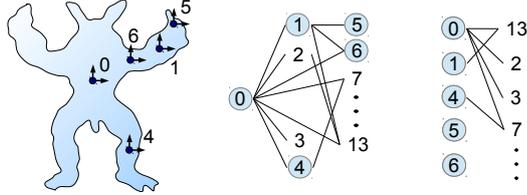


Figure 6: Mechanical hierarchy. Left: active nodes 0, 1, 4, 5, 6 at a given time. Middle: Reference hierarchy; nodes 2, 3, 7 are activation candidates; nodes 4, 5, 6 are deactivation candidates. Right: the resulting two-level contracted graph to be used in the mechanical simulation.

#### 377 4.2. Adaptive Hierarchy

378 In principle, we could start with an unstructured fine node  
 379 discretization of the objects and at each time step, find the best  
 380 simplifications by considering all possible deactivation and  
 381 activation candidates. However, in order to avoid a quadratic  
 382 number of tests, we pre-compute a node hierarchy and define  
 383 candidate nodes to be the ones at the interface between passive  
 384 and active nodes in the hierarchy.

##### 385 4.2.1. Hierarchy Setup

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

##### 397 4.2.2. Hierarchy Update

398 The candidates for activation  $\mathcal{AC}$  are the passive nodes with  
 399 all parents active. Conversely, the candidates for deactivation  
 400  $\mathcal{PC}$  are the active nodes with all children passive, if any, except  
 401 for the root of the reference hierarchy. In particular, active leaf  
 402 nodes are always in  $\mathcal{PC}$ . In the example shown in Fig. 6, nodes  
 403 4, 5, 6, 1, and 0 shown in the character outline are active. As  
 404 such, they do not mechanically depend on their parents in the  
 405 reference hierarchy, and the mechanical dependency graph is  
 406 obtained by removing the corresponding edges from the refer-  
 407 ence hierarchy. For edges in this two-levels graph, weights are  
 408 obtained by contracting the reference hierarchy using Eq. (5).

409 Similarly, two-level graphs can be obtained for the cases 1 and  
 410 3 discussed in the beginning of this section.

## 411 5. Adaptive Spatial Integration

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

### 414 5.1. Discretization

415 The spatial integration of energy and forces is numerically  
 416 computed using Gaussian quadrature, a weighted sum of values  
 417 computed at integration points. Exact quadrature rules are only  
 418 available for polyhedral domains with polynomial shape func-  
 419 tions (e.g. tri-linear hexahedra). In meshless simulation, such  
 420 rules do not exist in general. However, in linear blend skinning  
 421 one can easily show that the deformation gradient is uniform  
 422 (respectively linear) in regions where the shape functions are  
 423 constant (respectively linear). As studied in [7], uniform shape  
 424 functions can be only obtained with one node, so linear shape  
 425 functions between nodes are the best choice for homogeneous  
 426 parts of the material, since the interpolation then corresponds  
 427 to the solution of static equilibrium. One integration point of  
 428 a certain degree (*i.e.* one elaston [4]) is sufficient to exactly in-  
 429 tegrate polynomial functions of the deformation gradient there,  
 430 such as deformation energy in linear tetrahedra. We leverage  
 431 this property to optimize our distribution of integration points.  
 432 In a region  $\mathcal{V}_e$  centered on point  $\bar{\mathbf{p}}_e$ , the integral of a function  $g$   
 433 is given by:

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

434 where  $\mathbf{g}$  is a vector containing  $g$  and its spatial derivatives  
 435 up to degree  $n$  evaluated at  $\bar{\mathbf{p}}_e$ , while  $\mathbf{p}^{(n)}$  denotes a vector of  
 436 polynomials of degree  $n$  in the coordinates of  $\mathbf{p}$ , and  $\bar{\mathbf{g}}_e$  is a vec-  
 437 tor of polynomials integrated across  $\mathcal{V}_e$  which can be computed  
 438 at initialization time by looping over the voxels of an arbitrar-  
 439 ily fine rasterization. The approximation of Eq. (15) is exact if  
 440  $n$  is the polynomial degree of  $g$ . Due to a possibly large num-  
 441 ber of polynomial factors, we limit our approximation to quartic  
 442 functions with respect to material coordinates, corresponding to  
 443 strain energies and forces when shape functions are linear and  
 444 the strain measure quadratic (*i.e.* Green-Lagrangian strain).

445 Since the integration error is related to the linearity of shape  
 446 functions, we decompose the objects into regions of as linear as  
 447 possible shape functions at initial time, as shown in Fig. 7a and  
 448 Fig. 7b.

449 We compute the regions influenced by the same set of in-  
 450 dependent nodes, and we recursively split these regions until  
 451 a given linearity threshold is reached, based on the error of a  
 452 least squares linear fit of the shape functions. Let  $\phi_i(\bar{\mathbf{p}})$  be the  
 453 shape function of node  $i$  as defined in Eq. (1), and  $\mathbf{e}_i^T \bar{\mathbf{p}}^{(1)}$  its  
 454 first order polynomial approximation in  $\mathcal{V}_e$ . The linearity error  
 455 is given by:

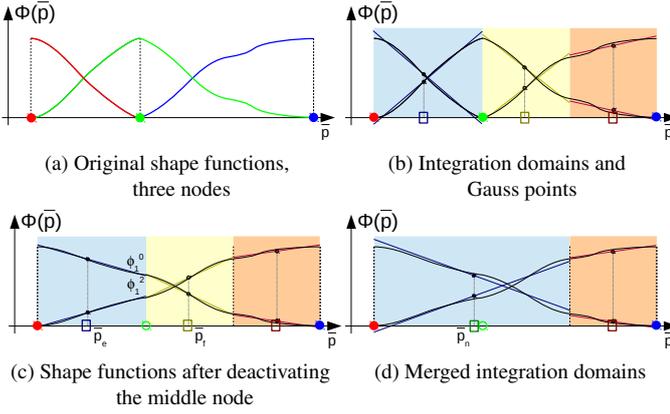


Figure 7: Adaptive integration points in 1D. Disks denote control nodes while rectangles denote integration points.

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

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

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

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

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

We solve for the least squares coefficients  $\mathbf{c}_i^e$  minimizing  $\varepsilon$ :  $\mathbf{c}_i^e = (A^e)^{-1} B_i^e$ . 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  $\bar{\mathbf{p}}_n$ . 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 is visually very close to the non-adaptive one.

### 6.2. Complex Scenes

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

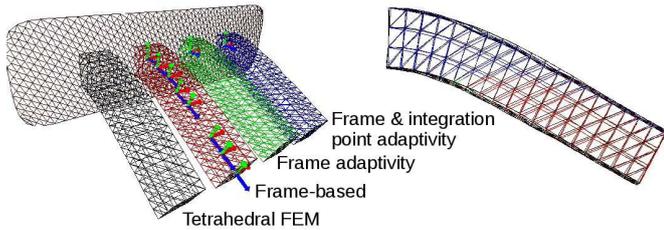


Figure 8: Four cantilever beams at equilibrium with the same properties and loading (fixed on one side, and subject to gravity). Right: perspective side view of the same configuration, showing that our adaptivity framework produces results similar (up to the depth-buffer precision, hence the color changes)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

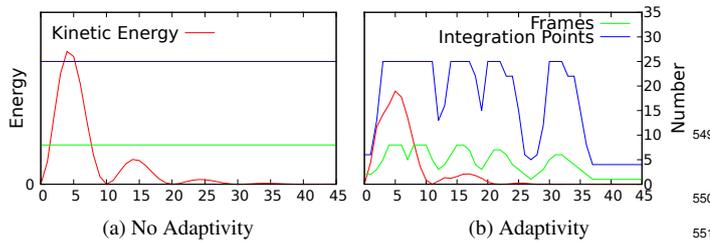


Figure 9: Kinetic Energy (red) Analysis with varying number of frames (green) and integration points (blue) over time (cantilever beam under flexion).

### 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.

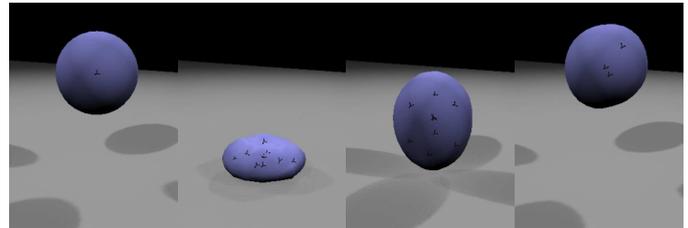


Figure 11: A falling deformable ball with unilateral contacts.

### 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

570 freely falling with air damping, and one node is sufficient to  
 571 simulate all of them.

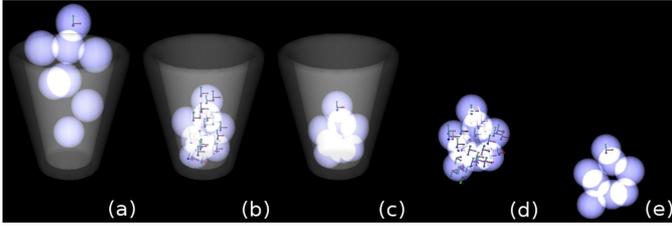


Figure 13: Eight deformable balls stacking up in a glass, which is eventually removed.

### 572 6.2.6. Armadillo Salad

573 A set of Armadillos (Fig. 12b) is dropped into a bowl, demon-  
 574 strating the scalability and robustness of our method in a diffi-  
 575 cult (self-)contacting situation.

### 576 6.3. Performance

577 In the various scenarios described above, our technique al-  
 578 lows a significant reduction of both kinematic DOFs and in-  
 579 tegration points, as presented in Table 2. Speedups are sub-  
 580 stantial, even when collision handling is time consuming. It is  
 581 worth noting that, for a fair comparison with the non-adaptive  
 582 case, our examples exhibit large, global and dynamical de-  
 583 formations.

584 In order to evaluate the gain of adaptivity regarding the  
 585 scene complexity, we throw armadillos in a bowl, at various re-  
 586 solutions. The speedups presented in Table 1 show that scenes  
 587 resulting in larger systems give better speedups since the com-  
 588 plexity of solving the system increases along with the number  
 589 of DOFs. The algorithmic complexity of solving deformable  
 590 object dynamics generally depends on three factors: the num-  
 591 ber of DOFs, the computation of elastic forces and, in the case  
 592 of iterative solvers, the conditioning of the system. By using  
 593 fewer integration points, our method is able to compute elastic  
 594 forces in a much faster way. In the case of badly conditioned  
 595 systems, as for instance tightly mechanically coupled system  
 596 (*e.g.* stacks), iterative methods need a large number of itera-  
 597 tions and thus the number of DOFs becomes critical. The de-  
 598 pendency on the number of DOFs is even larger when using di-  
 599 rect solvers. Thus, our method is particularly interesting in such  
 600 cases and allow for significant speedups compared to the non-  
 601 adaptive case. For instance: 6.25 $\times$  when the balls are stacked  
 602 into the glass (see Fig. 13c).

603 We noticed that the overhead due to adaptivity is moderate  
 604 compared to the overall computational time (typically between  
 605 5% and 10%), since adaptivity is incremental for both nodes  
 606 and integration points between two consecutive time steps. The  
 607 dense voxel grid is visited only once at initialization to compute  
 608 shape functions, masses, and integration data. Note that the  
 609 cost of our adaptivity scheme is independent from the method  
 610 to compute shape functions (they could be based on harmonic  
 611 coordinates, natural neighbor interpolants, etc).

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

Table 1: Speedups for a salad of one and 18 armadillos at various maximal resolutions (including collision timing)

### 612 6.4. Application to Mesh Registration

613 Using our adaptive scheme, the number of DOFs can be  
 614 progressively increased as needed during the registration pro-  
 615 cess, as shown in Fig. 14. Our approach is the following: we  
 616 start with a fully coarsened model and apply the deformable  
 617 registration procedure. Springs between the deformable sur-  
 618 face and corresponding points on the target surface are created  
 619 at each iteration. For a given spring stiffness, nodes become  
 620 active as needed to produce the required, increasingly local de-  
 621 formations, minimizing both external and internal energy. Af-  
 622 ter equilibrium is reached, our adaptive kinematic model comes  
 623 back to its fully coarsened state. When this happens, we pro-  
 624 gressively increase the registration spring stiffness, then again  
 625 let the adaptive model deform accordingly. We iterate this pro-  
 626 cess until the distance to the target surface becomes smaller  
 627 than a user-defined threshold. By using our adaptive kinemat-  
 628 ics, the deformation field seamlessly transitions from global to  
 629 local transformations as needed, implementing the coarse-to-  
 630 fine registration strategy. In addition, parts of the mesh that  
 631 are already registered will not be refined again as other parts  
 632 continue to undergo deformations, if these parts belong to sep-  
 633 arate branches of the node hierarchy. Consider for instance  
 634 the node hierarchy presented in Fig. 5: if the legs are already  
 635 registered (nodes 3, 4, 7, 8, 9, 10), the arm registration (nodes  
 636 1, 2, 5, 6, 11, 12) will not trigger leg node activation since these  
 637 nodes belong to separate subgraphs, thus avoiding unneces-  
 638 sary computations. A comparison between adaptive and non-  
 639 adaptive registration is presented in Fig. 15: our adaptive kin-  
 640 ematics produce better results even though the registration poten-  
 641 tial forces are the same in both cases. In this example, the over-  
 642 all computational time was about 3 minutes. Since most of it  
 643 was spent on closest point search, we did not notice significant  
 644 computational gains between the adaptive and non-adaptive cases.

### 645 7. Conclusion and Perspectives

646 We introduced a novel method for the run-time adaptivity  
 647 of elastic models. Our method requires few pre-processing (a  
 648 few seconds) contrary to existing model reduction techniques  
 649 based on modal analysis and system training. Nodes are sim-  
 650 plified as soon as their velocities can be described by nodes at  
 651 coarser levels of details, otherwise they are made independent.  
 652 Linear interpolation is particularly suited for linear materials  
 653 and affine deformations as it provides the static solution; there-  
 654 fore no refinement occurs except if inertia produces large ve-  
 655 locity gradients. In non-linear deformation such as bending and  
 656 twisting, new nodes are active to approximate the solution in  
 657 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.

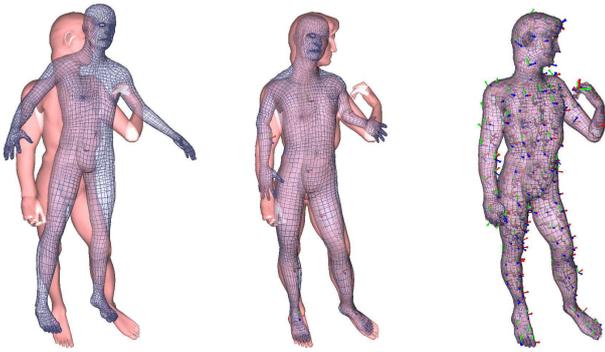


Figure 14: Deformable mesh registration of an adaptive deformable model (blue) to a target mesh (red). Left to right: while initially fully coarsened, the deformable model is progressively refined, automatically producing increasingly local deformations.

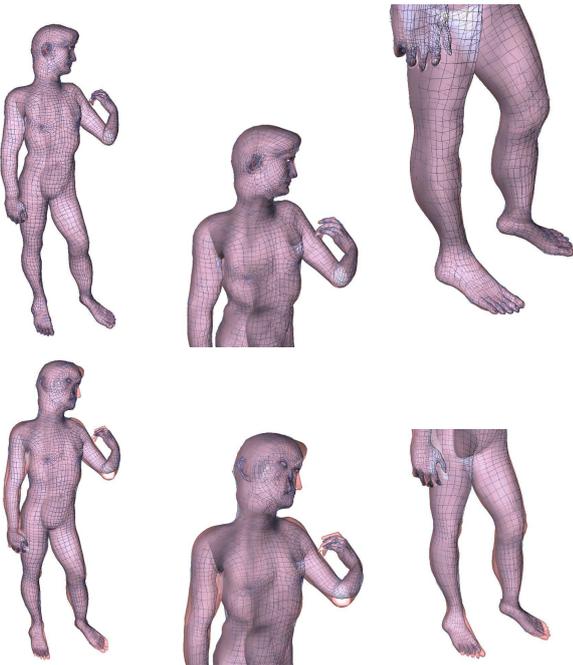


Figure 15: Comparison of deformable registration results for adaptive (top) and non-adaptive (bottom) kinematics. Our coarse-to-fine scheme produces a tighter fit, given the same registration potential forces.

ing 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–.

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). Reduc-

- 715 [13] Hildebrandt K, Schulz C, von Tycowicz C, Polthier K. Interactive space-  
716 time control of deformable objects. In: *ACM Transactions on Graphics*  
717 (TOG); vol. 31. ACM; 2012, p. 71–.
- 718 [14] Hutchinson D, Preston M, Hewitt T. Adaptive refinement for mass/spring  
719 simulations. In: *Eurographics Workshop on Computer Animation and*  
720 *Simulation*. 1996, p. 31–45.
- 721 [15] Ganovelli F, Cignoni P, Scopigno R. Introducing multiresolution repre-  
722 sentation in deformable object modeling. *ACM Spring Conference on*  
723 *Computer Graphics* 1999;.
- 724 [16] Delingette H, Cotin S, Ayache N. A hybrid elastic model allowing real-  
725 time cutting, deformations and force-feedback for surgery training and  
726 simulation. In: *Computer animation, 1999. Proceedings. IEEE; 1999*, p.  
727 70–81.
- 728 [17] Wu X, Downes MS, Goktekin T, Tendick F. Adaptive nonlinear fi-  
729 nite elements for deformable body simulation using dynamic progressive  
730 meshes. In: *Computer Graphics Forum; vol. 20. Wiley Online Library;*  
731 *2001*, p. 349–58.
- 732 [18] Debunne G, Desbrun M, Cani MP, Barr AH. Dynamic real-time deforma-  
733 tions using space & time adaptive sampling. In: *Proceedings of the*  
734 *28th annual conference on Computer graphics and interactive techniques.*  
735 *ACM; 2001*, p. 31–6.
- 736 [19] Grinspun E, Krysl P, Schröder P. Charms: a simple framework for adap-  
737 tive simulation. In: *ACM Transactions on Graphics (TOG); vol. 21.*  
738 *ACM; 2002*, p. 281–90.
- 739 [20] Sifakis E, Shinar T, Irving G, Fedkiw R. Hybrid simulation of deformable  
740 solids. In: *Proceedings of the 2007 ACM SIGGRAPH/Eurographics sym-*  
741 *posium on Computer animation. Eurographics Association; 2007*, p. 81–  
742 90.
- 743 [21] Martin S, Kaufmann P, Botsch M, Wicke M, Gross M. Polyhedral finite  
744 elements using harmonic basis functions. In: *Computer Graphics Forum;*  
745 *vol. 27. Wiley Online Library; 2008*, p. 1521–9.
- 746 [22] Lenoir J, Grisoni L, Chaillou C, Meseure P. Adaptive resolution of 1d  
747 mechanical b-spline. In: *Proceedings of the 3rd international conference*  
748 *on Computer graphics and interactive techniques in Australasia and South*  
749 *East Asia. ACM; 2005*, p. 395–403.
- 750 [23] Spillmann J, Teschner M. An adaptive contact model for the robust sim-  
751 ulation of knots. In: *Computer Graphics Forum; vol. 27. Wiley Online*  
752 *Library; 2008*, p. 497–506.
- 753 [24] Servin M, Lacoursiere C, Nordfelth F, Bodin K. Hybrid, multiresolution  
754 wires with massless frictional contacts. In: *Visualization and Computer*  
755 *Graphics, IEEE Transactions on; vol. 17. IEEE; 2011*, p. 970–82.
- 756 [25] Narain R, Samii A, O’Brien JF. Adaptive anisotropic remeshing for cloth  
757 simulation. In: *ACM Transactions on Graphics (TOG); vol. 31. ACM;*  
758 *2012*, p. 152–.
- 759 [26] Bargteil AW, Wojtan C, Hodgins JK, Turk G. A finite element method  
760 for animating large viscoplastic flow. In: *ACM Transactions on Graphics*  
761 (TOG); vol. 26. ACM; 2007, p. 16–.
- 762 [27] Wojtan C, Turk G. Fast viscoelastic behavior with thin features. In: *ACM*  
763 *Transactions on Graphics (TOG); vol. 27. ACM; 2008*, p. 47–.
- 764 [28] Wicke M, Ritchie D, Klingner BM, Burke S, Shewchuk JR, O’Brien JF.  
765 Dynamic local remeshing for elastoplastic simulation. In: *ACM Transac-*  
766 *tions on graphics (TOG); vol. 29. ACM; 2010*, p. 49–.
- 767 [29] Steinemann D, Otaduy MA, Gross M. Fast adaptive shape matching de-  
768 formations. In: *Proceedings of the 2008 ACM SIGGRAPH/Eurographics*  
769 *Symposium on Computer Animation. Eurographics Association; 2008*, p.  
770 87–94.
- 771 [30] Kim T, James DL. Skipping steps in deformable simulation with online  
772 model reduction. In: *ACM Transactions on Graphics (TOG); vol. 28.*  
773 *ACM; 2009*, p. 123–.
- 774 [31] Li H, Sumner R, Pauly M. Global correspondence optimization  
775 for nonrigid registration of depth scans. *Computer Graphics Forum*  
776 *2008;27(5):1421–30.*
- 777 [32] Allen B, Curless B, Popović Z. The space of human body shapes: Re-  
778 construction and parameterization from range scans. *ACM Trans Graph*  
779 *2003;22(3):587–94.*
- 780 [33] van Kaick O, Zhang H, Hamarneh G, Cohen-Or D. A survey on shape  
781 correspondence. *Computer Graphics Forum* 2011;30(6):1681–707.
- 782 [34] Besl P, McKay N. A method for registration of 3-d shapes. *IEEE Trans*  
783 *on Pattern Analysis and Machine Intelligence* 1992;14(2):239–56.
- 784 [35] Rusinkiewicz S, Levoy M. Efficient variants of the icp algorithm. *3-D*  
785 *Digital Imaging and Modeling* 2001;:145–52.
- 786 [36] Amberg B, Romdhani S, Vetter T. Optimal step nonrigid icp algorithms  
787 for surface registration. In: *CVPR’07. 2007*..
- 788 [37] Gilles B, Reveret L, Pai D. Creating and animating subject-specific  
789 anatomical models. *Computer Graphics Forum* 2010;29(8):2340–51.
- 790 [38] Magnenat-Thalmann N, Laperrrière R, Thalmann D. Joint dependent lo-  
791 cal deformations for hand animation and object grasping. In: *Graphics*  
792 *interface. 1988*, p. 26–33.
- 793 [39] Kavan L, Collins S, Žára J, O’Sullivan C. Skinning with dual quaternions.  
794 In: *Proceedings of the 2007 symposium on Interactive 3D graphics and*  
795 *games. ACM; 2007*, p. 39–46.
- 796 [40] Baraff D, Witkin A. Large steps in cloth simulation. In: *Proceedings of*  
797 *the 25th annual conference on Computer graphics and interactive tech-*  
798 *niques (SIGGRAPH). ACM; 1998*, p. 43–54.