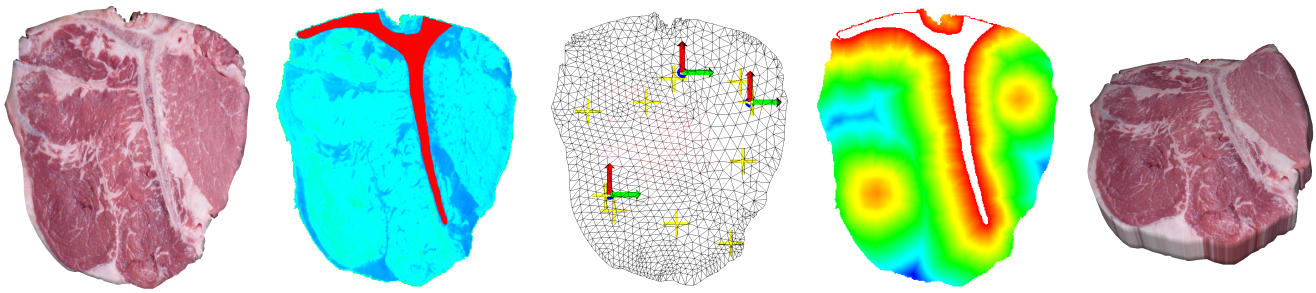
Sparse Meshless Models of Complex Deformable Solids

François Faure, Benjamin Gilles, Guillaume Bousquet, Dinesh K. Pai

Sparse Meshless Models of Complex Deformable Solids

François Faure[1,2,3] Benjamin Gilles[4,2,3] Guillaume Bousquet[1,2,3] Dinesh K. Pai[5]
[1]University of Grenoble [2]INRIA [3]LJK – CNRS [4]Tecnalia [5]UBC



(a) *T-Bone Steak* **(b)** *Stiffness* **(c)** *Discretization* **(d)** *Compliance distance* **(e)** *Deformation*

Figure 1: *A taste of the method. The T-bone steak (a) has a rigid bone and softer muscle and fat, as seen in the volumetric stiffness map (b). Our method can simulate it using only three moving frames and ten integration points (c), running at 500 Hz with implicit integration on an ordinary PC. The frame placement is automatically generated using a novel compliance-scaled distance (d). Observe that when one side of the meat is pulled (e), the bone remains rigid and the two meaty parts are correctly decoupled.*

Abstract

A new method to simulate deformable objects with heterogeneous material properties and complex geometries is presented. Given a volumetric map of the material properties and an arbitrary number of control nodes, a distribution of the nodes is computed automatically, as well as the associated shape functions. Reference frames attached to the nodes are used to apply skeleton subspace deformation across the volume of the objects. A continuum mechanics formulation is derived from the displacements and the material properties. We introduce novel material-aware shape functions in place of the traditional radial basis functions used in meshless frameworks. In contrast with previous approaches, these allow coarse deformation functions to efficiently resolve non-uniform stiffnesses. Complex models can thus be simulated at high frame rates using a small number of control nodes.

CR Categories: I.3.5 [Computer Graphics]: Physically based modeling— [I.3.7]: Computer Graphics—Animation

Keywords: physically based animation, deformable solid, meshless model

Links: DL PDF

1 Introduction

Physically based deformable models have become ubiquitous in computer graphics. So far, most of the work has focused on objects made of a single, homogeneous material. However, many real-world objects, including biological structures such as the T-bone steak shown in Figure 1, are composed of heterogeneous material. The simulation of such complex objects using the currently available techniques requires a high resolution spatial discretization to resolve the variations of material parameters. Consequently, the realistic simulation of such complex objects has remained impossible in interactive applications. In this paper, we address this question and we propose a novel approach to the simulation of heterogeneous, intricate materials with various stiffnesses.

The numerical simulation of continuous deformable objects is based on a discrete number of independent degrees of freedom (DOFs) which we will call *nodes* in this paper. Nodes can be control points, rigid frames, affine frames or any other primitive. Nodes are associated with *kernel functions* or *shape functions* (which are similar to kernels but normalized to give a partition of unity), which are combined to produce the *displacement function* of material points in the solid. In traditional methods, shape functions are geometrically designed to achieve a certain degree of locality and smoothness, independent of the material. The resulting deformations are rather homogeneous between the nodes. Accurately handling heterogeneous objects requires nodes at the boundaries between the different materials, and this raises well-known segmentation and meshing issues. While a small number of smooth material discontinuities may be tractable, handling multiple geometrically detailed boundaries requires a dense mechanical sampling that is incompatible with interactive simulation. Moreover, dense sampling creates numerical conditioning problems, especially in the case of stiff material.

In this paper, we show that it is possible to simulate complex heterogeneous objects with sparse sampling using new, material-aware shape functions. Our approach is based on a simple observation: points connected by stiff material move more similarly than connected by compliant material. The limit case is the rigid body, where all points move along with one single frame. We propose a meshless framework with a moving frame attached to each node,

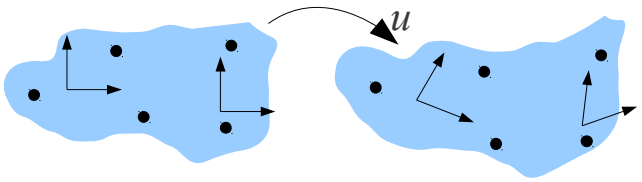


Figure 2: *The displacement u of a deformable object (blue) is discretized using nodes (black circles). Strain is measured based on the deformation of local frames (black arrows) computed at integration points in the material.*

and we show that carefully designing the kernel functions associated with the nodes allows a unification of rigid and deformable solid mechanics. This allows the straightforward modeling of heterogeneous objects and alleviates the meshing issues. Given a deformable object to simulate and a number of control nodes corresponding to an expected computation time, optimization criteria can be used to compute, at initialization time, a discretization of the object and the associated shape functions, in order to achieve a good realism.

Our specific contributions are the following: (1) a meshless simulation method for solids with material-aware shape functions using a novel distance function based on compliance; (2) a method to automatically model a complex object for this method, with an arbitrary number of sampling frames, based on surface meshes or volumetric data; (3) a system that implements the above methods and shows the ability to simulate complex deformation with a small number of dynamic degrees of freedom.

The remainder of this paper is organized as follows. We first briefly review in Section 2 relevant previous work, which allows us to motivate and sketch our approach with respect to the existing ones. In Section 3, we compare different types of shape functions and explain our choice. We then study in Section 4 the problem of material-aware shape functions starting in one dimension and extending to two or three dimensions, and propose a method to optimize the distribution of nodes. In Section 5, we derive the differential equation which governs the dynamics of the object, investigate precision issues and propose a strategy to optimize the distribution of integration points. We finally present results and discuss future work.

2 Background and motivation

The physical simulation of deformable solids has attracted a lot of attention in the recent decades, and a good survey can be found in [Nealen et al. 2005]. The displacement is typically sampled at a discrete set of points, which we call nodes in this document, and it is interpolated within the object based on the nodal values, as illustrated in Figure 2. Elastic forces can be derived using a classical hyperelastic scheme. Consider a material whose undeformed positions $\bar{\mathbf{p}}(\theta)$ are parametrized by local, curvilinear coordinates θ (like texture coordinates). When the material undergoes a deformation, the points are displaced to new positions $\mathbf{p}(\theta) = \bar{\mathbf{p}}(\theta) + \mathbf{u}(\theta)$, as illustrated in Figure 2. At each material point, the derivatives of the position function $\mathbf{p}$ with respect to the coordinates θ are the vectors of a local basis, illustrated using frames in Figure 2, and they are the columns of a 3×3 matrix F loosely called the deformation gradient, with reference value $\bar{F}$, typically the identity. The local deformation of the material is the non-rigid part of the transformation $\bar{F}^{-1}F$ between the reference and current states (like the distortion of a checkerboard texture). The strain, ϵ, is a measure of this deformation. Different strain functions have been used, but all of them fit in this derivation. The elastic response $\sigma(\epsilon)$ generates the elastic forces, and is a physical characteristic of the material. Its product with the strain is homogeneous to a density of energy. The elastic energy of a deformed object is the work done by the elastic forces from the undeformed state to the current state, integrated across the whole object: $\mathcal{W} = \int_{\mathcal{V}} \int_0^{\epsilon} \sigma d\epsilon$. Let $\mathbf{q}$ be the vector of independent DOFs of the object, responsible for the displacements $\mathbf{u}$. The associated elastic forces $\mathbf{f}$ are computed by differentiating the energy with respect to the DOFs:

$$\mathbf{f}^T = -\frac{\partial \mathcal{W}}{\partial \mathbf{q}} = -\int_{\mathcal{V}} \frac{\partial \epsilon}{\partial \mathbf{q}} \sigma \tag{1}$$

One additional differentiation provides us with the stiffness $\frac{\partial \mathbf{f}}{\partial q}$, used in implicit integration schemes and static solvers. Iterative linear solvers like the conjugate gradient only address the matrix through its product with a vector, which amounts to computing the change of force $\delta(\mathbf{f})$ corresponding to an infinitesimal change of position $\delta(\mathbf{q})$. This frees us from explicitly computing the stiffness matrix, and allows us to simply compute the changes of the terms in the force expression and accumulate their contributions:

$$\begin{aligned} \delta(\mathbf{f}^T) \quad = \quad & -\int_{\mathcal{V}} \frac{\partial \epsilon}{\partial \mathbf{q}} \frac{\partial \sigma}{\partial \epsilon} \frac{\partial \epsilon}{\partial \mathbf{q}}^T \delta(\mathbf{q}) \\ & -\int_{\mathcal{V}} \delta(\frac{\partial \epsilon}{\partial \mathbf{q}}) \sigma \end{aligned} \tag{2}$$

The first term corresponds to the change of stress intensity. The second corresponds to a change of direction due to non-linearity, and may be null or negligible, depending on the interpolation and strain functions. Damping forces, based on velocity, can straightforwardly be derived in this framework and added to the elastic forces.

Recently, precomputed deformation modes have been used to interactively deform large structures [James and Pai 2003; Barbič and James 2005; Kim and James 2009]. Using deformation modes rather than point-like nodes as DOFs allows to easily trade-off accuracy for speed. A layered model combining articulated body dynamics and a reduced basis of body deformation is presented in [Galoppo et al. 2009]. However, the deformation modes lack locality and pushing on one point may deform the whole object. Robustness problems such as inverted tetrahedra [Irving et al. 2006] or hourglass deformation modes in hexahedra [Nadler and Rubin 2003] have been addressed. To reduce computation time, embedding detailed objects in coarse meshes has become popular [Müller and Gross 2004; Sifakis et al. 2007; Nesme et al. 2009]. Disconnected or arbitrarily-shaped elements [Kaufmann et al. 2008; Martin et al. 2008] have been proposed to alleviate the meshing difficulties. Meshless methods, first introduced for fluid simulation, have been extended to solid mechanics [Müller et al. 2004; Gross and Pfister 2007]. Besides continuum mechanics-based methods, fast algorithms have been developed for video games to simulate quasi-isometry [Adams et al. 2008; Müller et al. 2005]. They are not able to model physical properties of materials, being based on geometry only.

Various displacement functions, strain measures and stress-strain laws have been proposed in the literature to cope with a wide range of object shapes and materials, small or large displacements, and numerical time integration methods.

In this work, we focus on the design of displacement functions. Our method is compatible with all strain measures, material parameters and integration methods. We briefly discuss the main classes of interpolation methods previously proposed. The seminal work of Terzopoulos et al.[1987] applied finite differences in a regular grid. In the more flexible Finite Element method [Bathe 1996], the displacement $\mathbf{u}$ is interpolated within the mesh cells (elements), as illustrated in Figure 3(a), based on cell vertices (nodes), as a

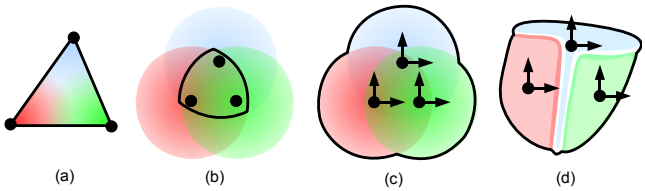


Figure 3: *Comparison of displacement functions. The black line encloses the area where the displacement function is defined, based on node positions (black circles) and associated functions (colored areas). (a) Finite Element, (b) Point-based, (c) Frame-based with RBF kernels, (d) Frame-based with our material-based kernels.*

weighted sum of the node displacements: $\mathbf{p}(\theta) = \sum_i w_i(\theta)\mathbf{p}_i$, where the $\mathbf{p}_i$ are the node positions, and the w are called the shape functions of the nodes. After years of constant progress, meshing and remeshing remains a difficult issue, see e.g. [Tournois et al. 2009]. While detailed geometry can be embedded within coarse cells to reduce complexity, detailed material requires a large number of cells, or complex precomputations at initialization time based on a finer mesh [Nesme et al. 2009; Kharevych et al. 2009]. To ease topological changes, point-based (also called meshless or meshfree) methods use radial basis functions attached to the nodes [Fries and Matthies 2003; Müller et al. 2004]. Sampling issues remain, since each interpolated point must lie in the range of at least four non-coplanar nodes, as illustrated in Figure 3(b). A very interesting meshless approach using moving frames was recently proposed to alleviate this limitation [Martin et al. 2010], using the generalized moving least squares (GMLS) interpolation. In this method, even one single neighboring node is sufficient to compute a local displacement, as illustrated in Figure 3(c). Moreover, the authors introduce a new affine (first-degree) approximation of the strain, called elaston. In contrast with the plain (zero-degree) strain value traditionally used, this allows each integration point to capture bending and twisting in addition to the usual stretch and shear modes. These improvements over previous methods remove all constraints on node neighborhood and allow the simulation of objects with arbitrary topology within a unified framework. However, a dense sampling of the objects is applied, leading to high computation times. Modeling a T-bone steak like the one depicted in Figure 1 using the meshless methods discussed above requires a large number of small-range nodes to prevent nodes on one side of the thin bone to influence the nodes on the other side. Moreover, the simulation of the rigid part requires highly stiff interaction forces which may generate numerical artifacts due to poor conditioning.

Our method overcomes these limitations, while keeping the benefits of the frame-based meshless framework. Sparse frame-based models were introduced in [Gilles et al. 2011], using skeleton subspace deformation (SSD) across the object volume and continuum mechanics to achieve physical realism. The weights are computed based on object geometry and frame distribution, independently of the material parameters, but it is shown that high stiffness can be simulated using appropriate shape functions. In this article, we show that SSD create more realistic deformations than GMLS for sparse models, and we present a new method to automatically distribute the nodes and compute the shape functions based on the material property map.

3 Sparse meshless models using skinning

To model deformable objects using a small number of control frames as discussed above, we need convenient, natural deformation functions. We use a generalization of the popular linear blend skinning method or Skeleton Subspace Deformation (SSD) [Magnenat-Thalmann et al. 1988], created to embed an articulated skeleton within a smooth deformable skin. In this section, we compare it with the most popular shape functions used in simulation and explain our choice. The displacements of control nodes (i.e., the DOFs q_i) are locally combined according to their shape function w_i, also called *weight*. The following derivations hold for different types of control nodes: points, rigid frames, linearly deformable (affine) frames, and quadratic frames. Let $\bar{\mathbf{p}}$ and $\mathbf{p}$ be positions in the initial and deformed settings and $\mathbf{u} = (\mathbf{p} - \bar{\mathbf{p}})$ the corresponding displacement, expressed as:

$$\mathbf{u} = \sum_i w_i(\bar{\mathbf{p}})\mathbf{A}_i\bar{\mathbf{p}}^* - \bar{\mathbf{p}} \ , \tag{3}$$

where $(\bar{\mathbf{p}})^*$ denotes a vector of polynomials of dimension d in the coordinates of $\bar{\mathbf{p}}$ and $\mathbf{A}_i(q_i)$ is a $3 \times d$ matrix. Positions in the reference, undeformed configuration are denoted using an upper bar. The matrix $\mathbf{A}_i$ represents the primitive transformation from its initial to its current position and is straightforwardly computed based on the independent DOFs: for instance, the 12 DOFs of an affine primitive are directly pasted into a 3×4 matrix, while the 6 DOFs of a rigid primitive are converted to a matrix using Rodrigues' formula. For point, affine or rigid, and quadratic primitives, we respectively use complete polynomial bases of order $n = 0$, $n = 1$, $n = 2$, noted as $(.)^n$. In 3D, we have $d = (n+1)(n+2)(n+3)/6$ and the three first bases are: $\mathbf{p}^0 = [1]$, $\mathbf{p}^1 = [1, x, y, z]^T$, $\mathbf{p}^2 = [1, x, y, z, x^2, y^2, z^2, xy, yz, zx]^T$. The weights constitute a partition of unity ($\sum w_i(\bar{\mathbf{p}}) = 1$) and can be computed by normalizing the kernel functions. To impose Dirichlet boundary conditions, it is convenient to have interpolating functions at $\bar{\mathbf{x}}_i$, the initial position (frame origin) of node q_i in 3d space: $w_i(\bar{\mathbf{x}}_i) = 1$ and $w_j(\bar{\mathbf{x}}_i) = 0, \forall j \neq i$.

Comparison with FEM and meshless methods: In finite element methods, the values of the shape function are typically the (generalized) barycentric coordinates of $\bar{\mathbf{p}}$ within the element which contains the point. It is easy to show that the linear (resp. quadratic) interpolation in a tetrahedron is equivalent to linear blend skinning with one affine (resp. quadratic) frame. Since shape functions are not bounded by elements, linear blend skinning can achieve a more global interpolation, which is more suitable with sparse DOFs.

In meshless methods, shape functions are computed based on kernel functions $\phi_i(\|\bar{\mathbf{p}} - \bar{\mathbf{x}}_i\|)$. In parametric models (e.g., splines), shape functions are generally polynomial functions of material coordinates. However, non-linear shape functions can lead to artifacts when modeling homogeneous materials with sparse primitives: in case of pure extension, a uniform deformation gradient $F = d\mathbf{p}/d\bar{\mathbf{p}}$ is expected, which occurs only if the shape functions are linear in $\bar{\mathbf{p}}$ (i.e., $dw_i/d\bar{\mathbf{p}}$ is uniform). Contrary to linear blend skinning (eq.3), the moving least squares (MLS) method [Fries and Matthies 2003] approximates the displacement by $\mathbf{u} = \mathbf{A}(\bar{\mathbf{p}})\bar{\mathbf{p}}^* - \bar{\mathbf{p}}$, where the $3 \times d$ blending matrix $\mathbf{A}(\bar{\mathbf{p}})$ is computed at initialization time by minimizing the interpolation error at primitive locations. For a point $\mathbf{p}$, the error is locally weighted using the kernel function: $e = \sum_i \phi_i(\|\bar{\mathbf{p}} - \bar{\mathbf{x}}_i\|)\|\mathbf{A}(\bar{\mathbf{p}})\bar{\mathbf{x}}_i^* - \mathbf{A}_i\bar{\mathbf{x}}_i^*\|^2$. The least squares fit leads to the closed form solution:

$$\begin{aligned} \mathbf{u} &= \sum_i \phi_i(\|\bar{\mathbf{p}} - \bar{\mathbf{x}}_i\|)\mathbf{A}_i\bar{\mathbf{x}}_i^*\bar{\mathbf{x}}_i^{*T}\mathbf{G}^{-1}\bar{\mathbf{p}}^* - \bar{\mathbf{p}} \\ \mathbf{G} &= \sum_i \phi_i(\|\bar{\mathbf{p}} - \bar{\mathbf{x}}_i\|)\bar{\mathbf{x}}_i^*\bar{\mathbf{x}}_i^{*T} \end{aligned} \tag{4}$$

Equation 4 looks very similar to the blending equation 3, except that initial positions $\bar{\mathbf{p}}^*$ are altered with a constant, non-uniform matrix used to maximize the quality of the interpolation. One drawback of

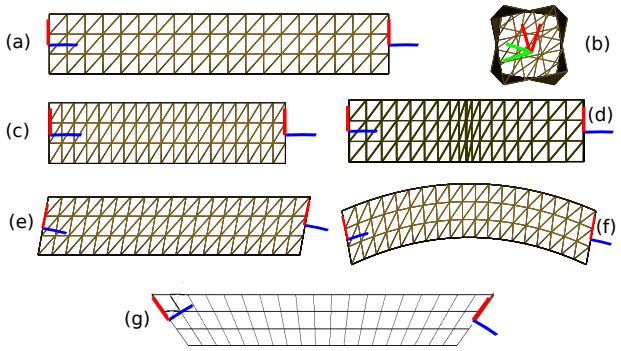


Figure 4: *Deformation modes. (a): rest shape, (b): twisting, (c),(d): compression with linear (resp. nonlinear) shape functions, (e): shear, (f): bending can be obtained using skinning, but (g): not using GMLS with linear weights.*

MLS is that $\mathbf{G}$ is invertible only if the point is weighted by at least d non-coplanar nodes. To remedy this, generalized MLS has been developed [Fries and Matthies 2003; Martin et al. 2010]. Contrary to linear blend skinning, the uniformity of deformations is not ensured in (G)MLS approaches since F is non linear in $\bar{\mathbf{p}}$ due to the rational function $\mathbf{G}^{-1}$. Moreover, using barycentric coordinates as kernel functions to maximize linearity, GMLS performs linear interpolations as illustrated in Figure 4, instead of the smooth, natural bending provided by skinning. This may be sufficient with dense sampling, since bending can be approximated by a sequence of affine transforms. In sparse models, the ability to simulate all the natural deformations between two nodes is highly desirable, which makes skinning a better choice. By combining skinning and elastons, a visually pleasing and physically sound model of a deformable solid with stretch, shear, as well as bending and twisting, can be obtained from only two frames and a single integration point.

4 Material-aware shape functions

Building sparse frame-based physical models not only requires appropriate deformation functions as discussed in the previous section, but also anisotropic shape functions to resolve heterogeneous material as illustrated in Figure 3(d). In this section, we propose a method to automatically compute such functions.

4.1 Compliance distance

Consider the deformation of a heterogeneous bar in one dimension, as shown in Figure 5, where each point $\mathbf{p}$ is parameterized by one material coordinate x. Let the endpoints $\mathbf{p}_0$ and $\mathbf{p}_1$ be the sampling points of the displacement field. At any point, the displacement is a weighted sum of the displacements at the sampling points: $\mathbf{u}(x) = w_0(x)\mathbf{u}_0 + w_1(x)\mathbf{u}_1$. If the bar is heterogeneous, the deformation is not uniform and depends on the local stiffness, as illustrated in Figure 5(b). We call a shape function *ideal* if it encodes the exact displacement within the bar given the displacements of the endpoints, as computed by a static solution. Choosing the static solution as the reference is somehow arbitrary, since inertial effects play a role in dynamics simulation. However, the computation of interior positions based on boundary positions is an ill-posed problem in dynamics, since the solution depends on the velocities and on the time step. Moreover, for graphics, we believe that our perception of realism is more accurate for static scenes than when the object is moving. Using the static solution as a shape function makes sense from this point of view, and encodes more information

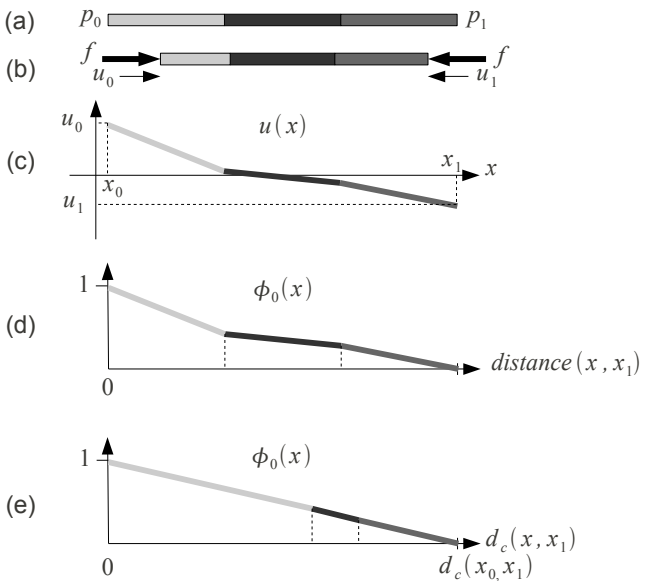


Figure 5: *Shape function based on compliance distance. (a): A bar made of 3 different materials, in rest state, with stiffness proportional to darkness. (b): The bar compressed by an external force. (c): The displacement across the bar. (d): The ideal w_0 shape function to encode the material stiffness. (e): The same, as a function of the compliance distance.*

than a purely geometric shape function.

It is possible to derive the ideal shape functions by computing the static solution $\mathbf{u}(x)$ corresponding to a compression force $\mathbf{f}$ applied to the endpoints. Note that this precomputation is exact for linear materials only. For simplicity, we assume that the bar has a unit section. At any point the local compression is $\epsilon = \frac{d\mathbf{u}}{dx} = \mathbf{f}/E = \mathbf{f}c$, where E is the Young's modulus, and its inverse c is the compliance of the material. Solving this differential equation provides us with: $\mathbf{u}(x) = \mathbf{u}(x_0) + \int_{x_0}^{x} \mathbf{f}c\ dx$, and since the force is constant across the bar, the shape function w_0 illustrated in Figure 5(d) is exactly:

$$w_0(x) = \frac{\mathbf{u}(x) - \mathbf{u}(x_1)}{\mathbf{u}(x_0) - \mathbf{u}(x_1)} = \frac{\int_x^{x_1} c\ dx}{\int_{x_0}^{x_1} c\ dx} \tag{5}$$

Let us define the compliance distance between two points $\mathbf{a}$ and $\mathbf{b}$ as: $d_c(a,b) = \int_{x_a}^{x_b} c\ |dx|$. The slope of the ideal shape function is: $\frac{dw_0}{dx} = -c/d_c(\mathbf{p}_0, \mathbf{p}_1)$. It is proportional to the local compliance c and to the inverse of the compliance distance between the endpoints. Interestingly, the shape function is thus an affine function of the compliance distance, as illustrated in Figure 5(e), and it can be computed without solving an equation.

4.2 Extension to two or three dimensions

We showed in the previous section that computing ideal shape functions in 1D objects, without performing compute-intensive static analyses as in [Nesme et al. 2009], is straightforward based on compliance distance. Let n be the number of points where we want to compute exact displacements to encode in shape functions. In one dimension, both the static solution and the distance field can be computed in linear time. In two dimensions, computing the static solution for n independent points requires the solution of a $2n \times 2n$ equation system. The worst case time complexity of the solution is $O(n^3)$, and direct sparse solvers can achieve it with a degree between 1.5 and 2 in practice. In contrast, the computation of an

approximate distance field in a voxel grid is $O(n \log n)$, which is much faster, but does not allow us to expect an exact solution like in one dimension. The reason is that there is an infinity of paths from one point to another to propagate forces across, thus the stress is not uniform and can not be factored out of the integrals and simplified like in Equation 5. Another difference with the one-dimensional case is the number of deformation modes. Higher-dimensional objects exhibit several stretching and shearing modes, and we can not expect the ratio of displacement between two points to be the same in each mode. Since a single scalar value can not encode several different ratios, there is no ideal shape function in more than one dimension. Another limitation of this measure is that the compliance distance is the length (compliance) of the shortest (stiffest) path from one point to the other, independently of the other paths. Thus, two points connected by a stiff straight sliver are at the same compliance distance as if they were embedded in a compact block of the same material, even though they are more rigidly bound in the latter case. Moreover, material anisotropy is not modeled using a scalar stiffness value. Nonetheless, the compliance distance allows the computation of efficient shape functions, as shown in the following.

4.3 Voronoi kernel functions

In meshless frameworks, each node is associated with a kernel function which defines its influence in space, as presented in Section 3. A wide variety of kernel functions have been proposed in the literature, most often based on spherical, ellipsoidal or parallelepipedal supports. Our design departs from this, and is guided by a set of properties that we consider desirable for the simulation of sparse deformable models. To correctly handle the example shown in Figure 3d, we need to restrict kernel overlap, to prevent the influence of the left node from unrealistically crossing the bone and reaching the flesh on the right. We thus need to constrain the kernel values, while keeping them as smooth as possible. In particular, we favor as-linear-as possible shape functions with respect to the compliance distance, in order to reproduce the theoretical solution in pure extension. A kernel value should not vanish before reaching the neighboring nodes, otherwise there would be a rigid layer around each node. With a sufficient number of radial basis functions, all the boundary conditions could be met [Powell 1990]. Unfortunately, shape functions computed with RBFs are generally global and can increase with distance, producing unrealistic deformations. Local RBFs have isotropic compact support, and are thus only approximating.

Since there is no general analytical solution that can satisfy all the desired properties, we numerically compute a discrete approximate solution on the voxelized material property map. Solving a Laplace or heat equation on the grid would require the solution of a large equation system, and would compute nonlinear weight functions.

A Voronoi partition of the volume allows us to easily compute kernels with compact supports, imposed values and linear decrease. This can be efficiently implemented in voxelized materials using Dijkstra's shortest path algorithm. Since a point on a Voronoi frontier is at equal distance from two nodes, we set the two kernel values to 0.5 at this point, and scale the distances accordingly inside each cell. To extend the distance function outside a cell, we generate the isosurface of kernel value $1/4$ by computing a new Voronoi surface between the $1/2$ isosurface and the other nodes. We can then recursively subdivide the intervals to generate a desired number of isosurfaces. The kernel values can straightforwardly be interpolated between the isosurfaces: for instance, the value at $\mathbf{P}_1$ in figure 6a is $(\frac{1}{2}d_{3/4} + \frac{3}{4}d_{1/2})/(d_{3/4} + d_{1/2})$, where d_i is the distance to isosurface of kernel value i, on Dijkstra's shortest path the point belongs to. To compute values between the last isosurface and 0 (the neighboring nodes), we apply a particular scheme since points beyond the neighbors, such as $\mathbf{P}_2$ in the figure, should not be influenced by the node. In this case, we linearly extrapolate the kernel function: $(-\frac{1}{2}d_{1/4} + \frac{1}{4}d_{1/2})/(d_{1/4} + d_{1/2})$. This technique is easily generalized to compliance distance and all the desired properties are met: it correctly generates interpolating, smooth, linear and decreasing functions between nodes. The corresponding cell shapes are not necessarily convex in Euclidean space, which allows them to resolve complex material distributions as illustrated by the compliance distance field in Figure 1d. Since points can be in the range of more than two kernels, a normalization is necessary to obtain a partition of unity and the linearity of the shape functions is not perfectly achieved, however the functions are often close to linear as shown in the accompanying video.

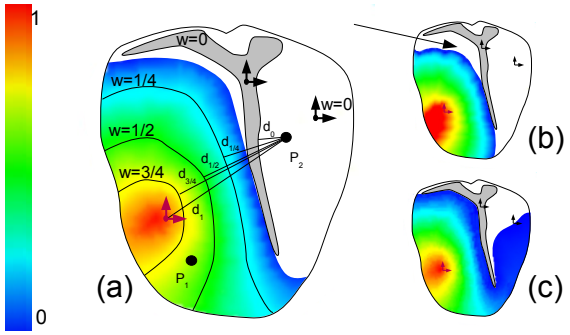


Figure 6: *Color map of the normalized shape function corresponding to the red node, computed using two Voronoi subdivisions (left), one subdivision (top right) and five subdivisions (bottom right).*

When using a small number of Voronoi subdivisions, we are not guaranteed to reach all the expected regions due to inaccurate extrapolation (see arrow tip in Figure 6b, where $d_1 < 2d_{1/2}$). Increasing the number of isosurfaces reduces this artifact, but can lead to unrealistically large influence regions as shown in Figure 6c, where the right part is influenced by the red node due to the linear interpolation between the right and left nodes. In practice, a small number of subdivisions are sufficient to remove noticeable artifacts while maintaining realistic bounds. The design of more realistic kernels in the extreme case of a very sparse discretization, large material inhomogeneities and complex geometry, is deferred to future work.

4.4 Node distribution

The Voronoi computations provide us with a natural way to uniformly distribute nodes in the space of compliance-scaled distances. We apply a standard farthest point sampling followed by a Lloyd relaxation (iterative repositioning of nodes in the center of their Voronoi regions) as done in [Adams et al. 2008; Martin et al. 2010]. The uniform sampling using the compliance distance results in higher node density in more compliant regions, allowing more deformation in soft regions. Since a whole rigid object corresponds to a single point in the compliance distance metric, all its points in Cartesian space have the same shape function values. Interestingly, it thus undergoes a rigid displacement, even if it is not associated to a single node. However, due to the well-known artifacts of linear blend skinning, it may actually undergo compression in case of large deformations. This artifact can be easily avoided by initializing nodes in the rigid parts, and keeping them fixed during the Lloyd relaxation. Another solution would be to replace linear blend skinning with dual quaternion skinning [Kavan et al. 2007], at the price of more complex mechanical computations due to normalization.

5 Space and time integration

This section explains how to set up the classical differential equation of dynamics for our models. The simulation loop is finally summarized in Algorithm 1.

5.1 Differential equation

We solve the classical dynamics equation

$$\mathbf{M}\ddot{\mathbf{q}} - \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{f}_{ext}(\mathbf{q}, \dot{\mathbf{q}}) \tag{6}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{q}$ and $\dot{\mathbf{q}}$ are the DOF value and rate vectors, $\ddot{\mathbf{q}}$ denotes the accelerations, $\mathbf{f}$ the internal forces, $\mathbf{f}_{ext}$ the external and inertial forces. Without loss of generality we consider Implicit Euler integration (see *e.g.*, [Baraff and Witkin 1998]), which computes velocity updates by solving the following equation:

$$\left(\mathbf{M} - h\mathbf{C} - h^2\mathbf{K}\right)\delta\dot{\mathbf{q}} = h\left(\mathbf{f}_{ext} + h\mathbf{K}\dot{\mathbf{q}}\right) \tag{7}$$

where h is the time step, $\mathbf{K} = \frac{\partial \mathbf{f}}{\partial \mathbf{q}}$ is the stiffness matrix, and $\mathbf{C} = \frac{\partial \mathbf{f}}{\partial \dot{\mathbf{q}}}$ the damping matrix, often represented using the popular Rayleigh assumption: $\mathbf{C} = \alpha\mathbf{M} + \beta\mathbf{K}$. The matrices need not to be explicitly computed, since the popular Conjugate Gradient solver addresses them only through their products with vectors. The generalized mass matrix is computed by assembling the $\mathbf{M}_{ij}$ blocks related to primitives i and j:

$$\mathbf{M}_{ij} = \int_{\mathcal{V}} \rho \frac{\partial \mathbf{u}}{\partial \mathbf{q}_i}^T \frac{\partial \mathbf{u}}{\partial \mathbf{q}_j}, \tag{8}$$

where ρ is the mass density. For simplicity, we lump the mass of each primitive by neglecting the cross terms : $\mathbf{M}_{ij} = 0, \forall i \neq j$. The resulting global mass matrix is block diagonal and the $\mathbf{M}_{ii}$ are square matrices, simplifying the time integration step without noticeable artifacts.

5.2 Elastic force

The computation of the elastic forces is explained in Section 2. However, we explicitly introduce the deformation gradient F in the force computation:

$$\mathbf{f}^T = -\frac{\partial \mathcal{W}}{\partial \mathbf{q}} = -\int_{\mathcal{V}} \frac{\partial \epsilon}{\partial \mathbf{q}}\sigma = -\int_{\mathcal{V}} \frac{\partial \epsilon}{\partial F}\frac{\partial F}{\partial \mathbf{q}}\sigma \tag{9}$$

This provides us with great modularity: the strain measure module computes $\frac{\partial \epsilon}{\partial F}$, while the interpolation module computes $\frac{\partial F}{\partial \mathbf{q}}$, and both can be designed and reused independently. A similar derivation is used to compute the product of the stiffness matrix with a vector:

$$\begin{aligned}\delta(\mathbf{f}^T) \quad = \quad & -\int_{\mathcal{V}} \tfrac{\partial \epsilon}{\partial F}\tfrac{\partial F}{\partial \mathbf{q}}\tfrac{\partial \sigma}{\partial \epsilon}\tfrac{\partial F}{\partial \mathbf{q}}^T\tfrac{\partial \epsilon}{\partial F}^T\delta(\mathbf{q}) \\ & -\int_{\mathcal{V}} \left(\delta(\tfrac{\partial \epsilon}{\partial F})\tfrac{\partial F}{\partial \mathbf{q}} + \tfrac{\partial \epsilon}{\partial F}\delta(\tfrac{\partial F}{\partial \mathbf{q}})\sigma\right),\end{aligned} \tag{10}$$

This modularity allows us to implement the blending of rigid, affine and quadratic primitives, and to easily combine them with a variety of strain measures. Note that other interpolation methods, such as FEM and particle-based methods, fit in this framework. We have implemented the popular corotational and Green-Lagrange strains, and Hookean material laws. Incompressibility is simply handled by measuring the change of volume, $\|F\| - 1$, and applying a scalar response using the bulk modulus. Other popular models such as Mooney-Rivlin and Arruda-Boyce would be easy to include.

5.3 Space integration

The quantities derived in the previous sections are numerically integrated across the material using a set of function evaluations. The accuracy of this process, called cubature, is described by its order, meaning that polynomial functions of lower degrees can be integrated exactly. As discussed in section 4, linear shape functions are desirable with sparse DOFs as they are able to generate uniform stretching. The following table summarizes the degrees of the different quantities obtained with linear shape functions for different strain measures and primitives. Classical cubature methods

Node	Strain measure	$\mathbf{u}$	$\mathbf{M}$	F	ϵ, σ	$\mathbf{f}$
Affine/Rigid	Corotational	2	4	1	1	2
Affine/Rigid	Green-Lagrange	2	4	1	2	4
Quadratic	Corotational	3	6	2	2	4
Quadratic	Green-Lagrange	3	6	2	4	8

Table 1: *Polynomial degrees obtained with linear shape functions.*

such as the midpoint rule (order 1), the Simpson's rule (order 3) or Gauss-Legendre cubature (order 5) would require many evaluation points to be accurate. The most representative evaluation points can be estimated as in [An et al. 2008], but it requires intensive static analysis at initialization time. Fortunately, displacements based on linear blend skinning can be easily differentiated and all quantities can be integrated explicitly in regions of linear weights. In a region $\mathcal{V}e$ centered on $\bar{\mathbf{p}}$ and containing points $\bar{\mathbf{p}} + d\bar{\mathbf{p}}$, an integrated quantity is $\int_{\mathcal{V}e} v = \mathbf{v}^T \int_{\mathcal{V}e} d\bar{\mathbf{p}}^n$ where $\mathbf{v}$ is a vector containing the quantity v and its spatial derivatives up to degree n, and $\int_{\mathcal{V}e} d\bar{\mathbf{p}}^n$ is the integrated polynomial basis of order n over the region. This last term can be estimated at initialization time using the voxel maps. This integration is exact if n is the polynomial degree of v. This formulation generalizes the concept of elastons [Martin et al. 2010] where quantities of order $n = 2$ are explicitly integrated in cuboid regions. Using $n = 0$, the integration scheme is equivalent to the midpoint rule: $\int_{\mathcal{V}e} v \approx v\mathcal{V}e$.

The method presented in section 4 generates as-linear-as possible shape functions. However, the gradients are discontinuous at the boundaries of the influence regions. We therefore partition the volume in regions influenced by the same set of nodes, and place one integration sample $\mathcal{V}e$ in each of them. To increase precision, we recursively subdivide the remaining regions up to the user-defined number of integration points. Our subdivision criterion is based on the error of a least squares fit of the voxel weights with a linear function. For more precision, stiffness changes could also be considered.

5.4 Visual and contact surfaces

Visual and contact surfaces can be attached to the deformable objects using the skinning method presented in section 3. These surfaces can also be used to generate the voxel data, as in the example shown in Figure 10. The compliance distances of the vertices are interpolated in the material volumetric map, and the shape functions are computed accordingly. Our framework sets no restriction on the collision detection and response methods. Any force $\mathbf{f}_\mathbf{p}$ applied to a point $\mathbf{p}$ on the contact surface can be accumulated in the control nodes using the following relation, deriving from the power conservation law:

$$\mathbf{f} += \frac{\partial \mathbf{p}}{\partial \mathbf{q}}^T \mathbf{f}_\mathbf{p} \tag{11}$$

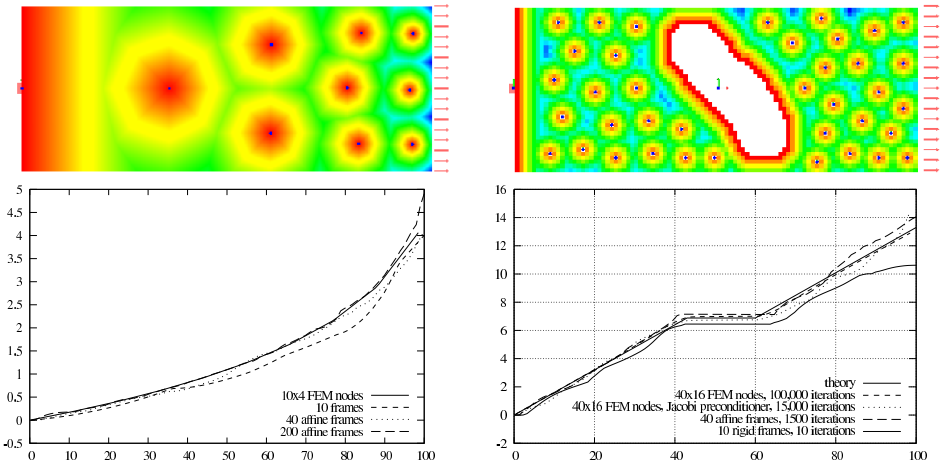


Figure 7: *Comparison with FEM on the extension of a plate. Left: with a stiffness gradient. Right: uniform stiffness and rigid part. Top: compliance distance field within each Voronoi cell, left:* 500×200 *grid, right:* 100×40 *grid*

Data: Voxel map of material properties, number of control nodes
Initialization:

- Distribute the control nodes ; `// sec. 4.4`
- Compute the shape functions ; `// sec. 4.3`
- Compute the mass matrix ; `// sec. 5.1`
- Generate the integration samples ; `// sec. 5.3`
- Compute the weights of the surface vertices ; `// sec. 5.4`

Loop:

- Accumulate force from each integration point: `// eq. 9`
  - Compute F (and its spatial derivatives);
  - Compute ϵ (and derivatives) from F using a given strain measure;
  - Compute σ (and derivatives) from ϵ using a given material model;
  - Add integrated force to each influencing primitive;
- External forces and collision handling ; `// eq. 11`
- At each solver iteration: `// eq. 7`
  - Accumulate force change from each integration point ; `// eq. 10`

Algorithm 1: Deformable model computations.

6 Results

6.1 Validation

We implemented our method within the SOFA framework [Allard et al. 2007] to exploit its implicit and static solvers, as well as its GPU collision detection and response [Allard et al. 2010]. To encourage its use, our software will be freely available in the upcoming release. We measured the displacement of the centerline of a 10×4 thin plate, as shown in Figure 7. The left side is fixed, while a uniform traction is applied to the right side. As expected, we obtain similar results using FEM and our method, with the same material parameters. A slight over-extension occurs in dense frame distributions, probably due to numerical issues in the voxel-wise integration of the deformation energy. We have also compared the simulations of cantilever beams under gravity, as illustrated in Fig. 8. The behaviors converge as we increase the number of nodes. These results were obtained using aligned nodes and analytically computed linear shape functions.

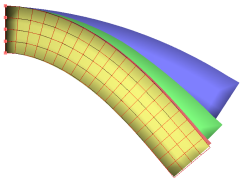


Figure 8: *Comparison of our models (solid colors) with FEM (wireframe). Blue: with two affine frames. Green: three affine frames. Red: five affine frames. Yellow: nine affine frames*

6.2 Performance

Computation times are difficult to compare rigorously because we use an iterative solver based on the conjugate gradient algorithm. In the test on heterogeneous material shown in Figure 7, we measured the total number of CG iterations applied to reach their final state with less than 1% of precision. The frame-based models converged from one to three orders of magnitude faster, thanks to the reduced number of DOFs. We used a regular FEM mesh. A more sophisticated meshing strategy taking the stiffness into account would certainly be more efficient, unfortunately implementations of these are not easily available. Carefully designed meshes can greatly enhance the speed of the FEM method. However, resolving geometrical details requires fine meshes with a large number of DOFs, and in case of large variations of stiffness, numerical issues considerably slow down the convergence, even using preconditioning. The ability of our method to encode the stiffness in the shape functions not only reduces the number of necessary DOFs, but also seems to reduce the conditioning problems. The pre-computation times range from less than one second for 10 frames in a 100×40 voxel grid to 10 minutes for 200 frames in a 500×200 grid. Our implementation is straightforward and there is plenty of room for optimization and parallelization.

Table 2 presents frame rates achieved on a common PC ($2.67Hz$ processor, $8GB$, Nvidia 295GTx). They include all the computations, including rendering and collision detection. The dragon and the ribbon demos are shown in the video. The computation times strongly depend on the number of integration points, which suggests that a GPU implementation of the force computations may dramatically increase the speed. A faster node relaxation [Liu et al. 2009] would speed up the precomputations in fine grids.

Corotational strain is about 1.5 times faster than Green-Lagrange strain due to the lower degree integration. However, in our implementation, it is not as robust because the rotation part of F is not differentiated to compute forces (eq. 10). Their accuracy on static solutions is comparable in our tests. Rigid and affine primitives exhibit similar computational time. Quadratic primitives are about 15% slower with the same number of integration points of

Model	# frames	# samples	# vertices	# voxels	t_{ini}	FPS
Steak	3	10	5k	$67k$	$3s$	500
Steak	10	53	5k	$67k$	$8s$	100
Steak	20	140	5k	$67k$	$12s$	40
Dragon	3	6	20k	$7M$	$100s$	300
Dragon	10	41	20k	$7M$	$220s$	150
Dragon	20	158	20k	$7M$	$360s$	27
Ribbon	5	9	4k	$60k$	$4s$	200
Knee	10	200	35k	$500k$	$11s$	10
Rat	30	230	600k	$1.5M$	$90s$	8

Table 2: *Timings.*

Figure 9: *The flesh and the fat, although interpolated between the same two control frames, pulled at the black point, exhibit different strains due to different stiffnesses.*

the same degree. We believe that the best compromise between accuracy and performance is achieved using affine primitives: they have more DOFs than rigid frames so can capture more deformation modes, and they require significantly fewer integration points than quadratic primitives if we limit the expansion of the deformation gradient to the first order, and the integration degree to 4 ($= 30$ polynomial terms). In theory, quadratic primitives would need a second order expansion of F and an 8^{th} order integration ($= 165$ polynomial terms), which would be more costly. Affine and rigid frames require only one integration point per region with linear shape function, providing the main deformation modes of a rod using only two frames and one integration point (see Figure 4). In our implementation, linear blend skinning of rigid frames is about five times faster than dual quaternion skinning [Gilles et al. 2011] with the same number of integration points. Dual quaternion skinning is more accurate in large bending (no volume loss) but requires more integration points due to the non-linear blending function and is significantly more complex to implement.

We found that sampling integration points in the overlapping influence regions was a suitable strategy, since it allowed a good linear approximation of the fine grained shape function defined in the voxel grid: in our test, the average difference was 0.05 (the shape function being defined between 0 and 1, making the approximation error less than 5%). This result also shows that the normalization of the kernel function does not significantly change the linearity. Uniformly distributed sample points unrealistically increase the stiffness because soft parts are assigned with a high stiffness due to averaging in the sample region.

6.3 Simulations

Figure 9 shows a close-up of the high speed simulation presented in Figure 1, which runs at haptic rates. The fat undergoes more deformation than the flesh because it is more compliant, even though they are interpolated between the two same control frames. The method of [Nesme et al. 2009] also realistically resolves heterogeneous materials, but a rigid bone across several elements would result in high stiffnesses and generate numerical problems. In contrast, our method handles the rigid parts straightforwardly, independently of their shape.

Our method allows the interactive simulation of complex biological systems such as the knee joint shown in Figure 10. In this model,

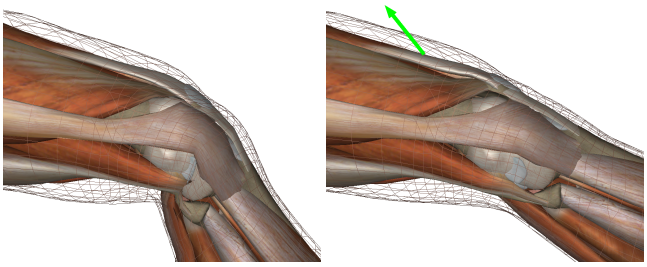


Figure 10: *Interactive knee simulation using* 10 *nodes. Pulling the quadriceps lifts the tibia.*

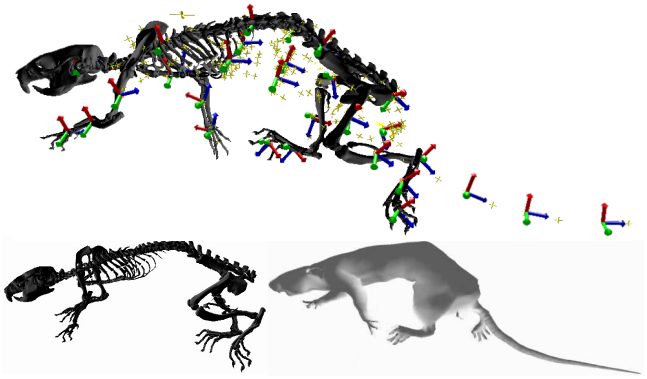


Figure 11: *Mix of animation and simulation. Top: the complete model. Bottom left: model subset animated using motion capture. Bottom right: result, with flesh and tail animated using physics.*

we have integrated four different tissues: bones, muscles, fat and ligaments. With only 10 nodes, we are able to realistically simulate flexion and fine movements such as the motion of the patella (knee cap) at 10 frames per second, without any prior knowledge of the kinematic skeleton. Forces are transmitted from the quadriceps to the tibia suggesting that accurate dynamic models of the anatomy, taking into account muscle actuation, could be built. A few modifications in the voxelization and material modules would allow motion discontinuity between tissues in contact and a more accurate simulation of the highly anisotropic non-linear fibrous biological tissues.

Our method can also be used to easily simulate physically-based secondary motions from skeleton-based animation or motion capture, as illustrated in Figure 11. The skin and the complete skeleton of a rat were acquired from micro-CT data. We applied our method to automatically sample the intermediate soft tissues and the tail with additional nodes. Optical motion capture was used to capture the movements of the limbs, head and three bones on the back. Real-time interaction with the soft tissues such as the belly and the tail is shown in the accompanying video.

7 Conclusion

We have introduced novel, anisotropic kernel functions using a new definition of distance based on compliance, which allow the encoding of detailed stiffness maps in coarse meshless models. They can be combined with the popular skinning deformation method to straightforwardly simulate large, physically sound deformations of complex heterogeneous objects, using small numbers of control nodes and small computation times. In contrast with classical FEM

and with the methods using geometrical shape functions, our approach decouples the resolution of the material from the resolution of the displacement function. The ability of setting an arbitrarily low number of frames, combined with a compliance-based distribution strategy, allows fast models to capture the most relevant deformation modes. In future work, we plan to investigate the dynamic adaptivity of the models allowed by the low pre-processing time of the method. Since sparse DOFs are not able to produce local deformations contrary to dense method such as FEM and GMLS, accuracy could be improved by refining contact regions on-the-fly.

Acknowledgements

We would like to thank Laurence Boissieux and François Jourdes for models, Estelle Duveau and Lionel Revéret for the rat data. This work is partly funded by European project *PASSPORT for Liver Surgery* (ICT-2007.5.3 223894) and French ANR project *SoHuSim*. Thanks to the support of the Canada Research Chairs Program, NSERC, CIHR, Human Frontier Science Program, and Peter Wall Institute for Advanced Studies.

References

ADAMS, B., OVSJANIKOV, M., WAND, M., SEIDEL, H.-P., AND GUIBAS, L. 2008. Meshless modeling of deformable shapes and their motion. In *Symposium on Computer Animation*, 77–86.

ALLARD, J., COTIN, S., FAURE, F., BENSOUSSAN, P.-J., POYER, F., DURIEZ, C., DELINGETTE, H., AND GRISONI, L. 2007. SOFA - an open source framework for medical simulation. In *Medicine Meets Virtual Reality, MMVR 15, 2007*, 1–6.

ALLARD, J., FAURE, F., COURTECUISSE, H., FALIPOU, F., DURIEZ, C., AND KRY, P. 2010. Volume contact constraints at arbitrary resolution. *ACM Transactions on Graphics 29*, 3.

AN, S. S., KIM, T., AND JAMES, D. L. 2008. Optimizing cubature for efficient integration of subspace deformations. *ACM Trans. Graph. 27*, 5, 1–10.

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. *SIGGRAPH Comput. Graph. 32*, 106–117.

BARBIČ, J., AND JAMES, D. L. 2005. Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Transactions on Graphics (SIGGRAPH 2005) 24*, 3 (Aug.), 982–990.

BATHE, K. 1996. *Finite Element Procedures*. Prentice Hall.

FRIES, T.-P., AND MATTHIES, H. 2003. Classification and overview of meshfree methods. Tech. rep., TU Brunswick, Germany.

GALOPPO, N., OTADUY, M. A., MOSS, W., SEWALL, J., CURTIS, S., AND LIN, M. C. 2009. Controlling deformable material with dynamic morph targets. In *Proc. of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*.

GILLES, B., BOUSQUET, G., FAURE, F., AND PAI, D. 2011. Frame-based elastic models. *ACM Transactions on Graphics*.

GROSS, M., AND PFISTER, H. 2007. *Point-Based Graphics*. Morgan Kaufmann.

IRVING, G., TERAN, J., AND FEDKIW, R. 2006. Tetrahedral and hexahedral invertible finite elements. *Graph. Models 68*, 2.

JAMES, D. L., AND PAI, D. K. 2003. Multiresolution green's function methods for interactive simulation of large-scale elastostatic objects. *ACM Trans. Graph. 22* (January), 47–82.

KAUFMANN, P., MARTIN, S., BOTSCH, M., AND GROSS, M. 2008. Flexible simulation of deformable models using discontinuous galerkin fem. In *Symposium on Computer Animation*.

KAVAN, COLLINS, ZARA, AND O'SULLIVAN. 2007. Skinning with dual quaternions. In *Symposium on Interactive 3D graphics and games*, 39–46.

KHAREVYCH, L., MULLEN, P., OWHADI, H., AND DESBRUN, M. 2009. Numerical coarsening of inhomogeneous elastic materials. *ACM Trans. Graph. 28* (July), 51:1–51:8.

KIM, T., AND JAMES, D. L. 2009. Skipping steps in deformable simulation with online model reduction. *ACM Trans. Graph. 28* (December), 123:1–123:9.

LIU, Y., WANG, W., LÉVY, B., SUN, F., YAN, D.-M., LU, L., AND YANG, C. 2009. On Centroidal Voronoi TessellationEnergy Smoothness and Fast Computation. *ACM Trans. Graph. 28* (08).

MAGNENAT-THALMANN, N., LAPERRIÈRE, R., AND THALMANN, D. 1988. Joint dependent local deformations for hand animation and object grasping. In *Graphics interface*, 26–33.

MARTIN, S., KAUFMANN, P., BOTSCH, M., WICKE, M., AND GROSS, M. 2008. Polyhedral finite elements using harmonic basis functions. *Comput. Graph. Forum 27*, 5.

MARTIN, S., KAUFMANN, P., BOTSCH, M., GRINSPUN, E., AND GROSS, M. 2010. Unified simulation of elastic rods, shells, and solids. *SIGGRAPH Comput. Graph. 29*, 3.

MÜLLER, M., AND GROSS, M. 2004. Interactive virtual materials. In *Graphics Interface*.

MÜLLER, M., KEISER, R., NEALEN, A., PAULY, M., GROSS, M., AND ALEXA, M. 2004. Point based animation of elastic, plastic and melting objects. In *Symposium on Computer Animation*, 141–151.

MÜLLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2005. Meshless deformations based on shape matching. *ACM Transactions on Graphics 24*, 3, 471–478.

NADLER, B., AND RUBIN, M. 2003. A new 3-d finite element for nonlinear elasticity using the theory of a cosserat point. *Int. J. of Solids and Struct. 40*, 4585–4614.

NEALEN, A., MÜLLER, M., KEISER, R., BOXERMAN, E., AND CARLSON, M. 2005. Physically based deformable models in computer graphics. In *Comput. Graph. Forum*, vol. 25 (4), 809–836.

NESME, M., KRY, P., JERABKOVA, L., AND FAURE, F. 2009. Preserving Topology and Elasticity for Embedded Deformable Models. *SIGGRAPH Comput. Graph.*.

POWELL, M. J. D. 1990. The theory of radial basis function approximation. *University Numerical Analysis Report*.

SIFAKIS, E., DER, K. G., AND FEDKIW, R. 2007. Arbitrary cutting of deformable tetrahedralized objects. In *Symposium on Computer Animation*.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. In *SIGGRAPH Comput. Graph.*, M. C. Stone, Ed., vol. 21, 205–214.

TOURNOIS, J., WORMSER, C., ALLIEZ, P., AND DESBRUN, M. 2009. Interleaving delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Trans. Graph. 28* (July), 75:1–75:9.