Chapter 1
Frame-based interactive simulation of complex deformable objects

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

Abstract We present a new type of deformable model which combines the realism of physically based continuum mechanics models and the usability of frame-based skinning methods, allowing the interactive simulation of objects with heterogeneous material properties and complex geometries. The degrees of freedom are coordinate frames. In contrast with traditional skinning, frame positions are not scripted but move in reaction to internal body forces. The deformation gradient and its derivatives are computed at each sample point of a deformed object and used in the equations of Lagrangian mechanics to achieve physical realism. We introduce novel material-aware shape functions in place of the traditional radial basis functions used in meshless frameworks, allowing 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.

1.1 Introduction

Deformable models are essential in mechanical engineering, biomechanics and computer graphics, typically for simulating the behavior of soft objects. The classical approach is *physically based deformation*, typically using continuum mechanics. This has the significant advantage of physical realism. Complex deformations are generated by numerical integration of discretized differential equations. However, these methods can be expensive and difficult to use. In the popular Finite Element Method (FEM) framework, the degrees of freedom of the discretized model are the vertices of a mesh, which must be constructed for each simulation object. A relatively fine mesh (i.e., a dense sampling of the deformation field) is required to capture common deformations such as torsion, leading to expensive simulations. Mesh adaptation can be difficult due to the topological constraints of the mesh. Particle-based meshless methods have been proposed to address these problems. While they obviate the need to maintain mesh topology, particles can not be placed arbitrarily. Therefore, these methods also need a dense cloud of particles not very different from the vertices of an FEM mesh.

Another approach, from the computer graphics community, is *skinning* (also known as vertex blending or skeletal subspace deformation). The deformation is kinematically generated by manipulating "bones," i.e., specific coordinate frames. This method is widely used, not only for its simplicity and efficiency, but because it provides natural and intuitive handles for controlling deformation. Skinning generates smooth deformations using a very sparse sampling of the deformation field. Adaptation is simple since frames can be inserted easily to control local features. These interesting features have made it the most widely used method for character animation. However, as a consequence of its purely kinematic nature (i.e., the frame positions need to be scripted), achieving physically realistic dynamic deformation is a major challenge with this approach.

We present a new approach that combines the advantages of both physically based deformation and skinning [13]. Instead of the vertices of a mesh, the degrees of freedom are a sparse set of coordinate frames. The equations of motion are derived for the moving frames by applying the principles of continuum mechanics across the volume of the deformed object, and solved using classical implicit time integration.

In addition, we show that it is possible to simulate complex heterogeneous objects with sparse sampling using new, material-aware shape functions [10]. So far, most of the work has focused on objects made of a single, homogeneous material. However, many real-world objects, including biological structures, 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. However, dense sampling creates numerical conditioning problems, especially in the case of stiff material. 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. Consequently, the realistic simulation of such complex objects has remained impossible in interactive applications. Our approach is based on a simple observation: points connected by stiff material move more similarly than connected by compliant material. 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 new approach which unifies skinning and physically based deformation modeling. (2) material-aware shape functions using a novel distance function based on compliance; (3) 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; (4) 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 chapter is organized as follows. We first briefly review in Section 1.2 relevant previous work, which allows us to motivate and sketch our approach with respect to the existing ones. In Section 1.3, we present the kinematic discretization using frames, and the interpolation function based on *skinning*. In Section 1.4, we derive the differential equation which governs the dynamics of the object, investigate precision issues and propose a strategy to optimize spatial integration. We then study in Section 1.5 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. We finally present results and discuss future work.

1.2 Related Work

Physically based deformable models have attracted continuous attention in Computer Graphics, since the seminal work of Terzopoulos [39]. We refer the reader to the excellent survey of [31] on this topic. Here, we briefly review the main Lagrangian models of deformable objects.

Mesh-based methods: Early works on deformable models in Computer Graphics have focused on interconnected particles. In mass-spring systems [35], constraints on edge length are enforced to counter stretching. Bending and shear can be controlled using additional springs. More general constraints such as area or volume conservation can be enforced using appropriate energy functions [40]. To realistically model volumetric deformable objects, it is necessary to apply continuum mechanics. The spatial derivatives of the displacement field can be computed using finite differences on a regular grid [39]. [38] studied the case of physically deformable NURBS surfaces for shape modeling. Finite elements [7, 14, 33, 8] allow irregular meshes, which are generally more convenient to sample objects with arbitrary shapes, but may be poorly conditioned. The spatial domain is subdivided into elements such as triangles, hexahedra or more frequently tetrahedra, in which the displacement field is interpolated using shape functions. At each point the strain can be computed using the spatial derivatives of the displacement field. Accurate material models have been implemented from rheological models relating stress and strain in hyperelastic, viscoelastic, inhomogeneous, transversely isotropic and/or quasi-incompressible media [41]. For simplicity, linearized strain has been applied assuming small displacements in rotated frames [29]. Precomputed deformations modes have been used to interactively deform large structures [18, 6, 22]. 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 [12]. However, the deformation modes lack locality and pushing on one point may deform the whole object. Models based on Cosserat points have been proposed for large deformations in thin structures [34] and solids [30]. Since robustness problems such as inverted tetrahedra [17] or hourglass deformation modes in hexahedra [30] have been addressed, meshing remain the main issue in finite elements. To reduce computation time, embedding detailed objects in coarse meshes has become popular in computer graphics [29, 37, 32]. Multi-resolution approaches have been proposed [9, 15]. In recent work, disconnected or arbitrarily-shaped elements [19, 26] have been proposed to alleviate the meshing difficulties.

Meshless methods: Meshless methods do not use an underlying embedding structure but unstructured control points. In computer graphics, meshless methods have been first introduced for fluid simulation and then extended to solid mechanics [28, 16]. Besides continuum mechanics-based methods, fast algorithms have been developed for video games to simulate quasi-isometry [1, 27]. They are not able to model real materials, being based on geometry only. In meshless methods, each control node has a given influence that generally decreases with the distance to it. Standard approximation or interpolation methods have been investigated for physical simulation such as Shepard functions, radial basis functions and moving least squares (see [11] for an extensive review). Despite the added flexibility due to the absence of elements, sampling issues remain, since each interpolated point must lie in the range of at least four non-coplanar nodes, as illustrated in Figure 1.1(b), contrary to our method that explicitly use rotations in the degrees of freedom. A very interesting meshless approach using moving frames was recently proposed to alleviate this limitation [25], 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 1.1(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.

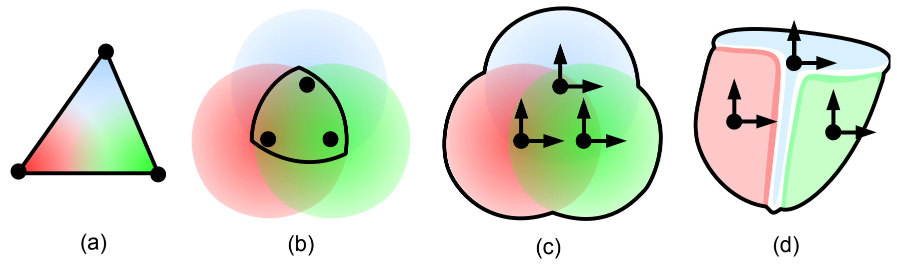


Fig. 1.1: Comparison of displacement functions. The black line encloses the area where the displacement function is defined, based on node positions (black circles) and associated functions (colored areas). (a) Finite Element, (b) Point-based, (c) Frame-based with RBF kernels, (d) Frame-based with our material-based kernels.

1.3 Frame-based deformation

In continuum media mechanics, it is necessary to numerically solve systems of differential equations (see Section 1.4). A general procedure is to smoothly approximate continuous functions in the solid from sought values at discrete sample locations. These values are the independent degrees of freedom (i.e., the DOFs q_i) which we will call *nodes*. In most simulation methods (Section 1.2), nodes are points and the deformation in the material is linearly interpolated from node displacements. In contrast, we consider rigid frames, affine frames and quadratic frames. Nodes are associated with *shape functions*, also called *weights*, which are combined to produce the displacement function of material points in the solid. To model deformable objects using a small number of control nodes, we need convenient, natural deformation functions. In character animation, the blending of frame displacements has been studied to deform a skin from an embedded articulated skeleton [21]. This method, called *skinning* or vertex blending or skeletal subspace deformation, is widely used, not only for its simplicity and efficiency, but because it provides natural and intuitive handles for controlling deformation. Skinning generates smooth deformations using a very sparse sampling of the deformation field. Here, we present two different blending techniques that we have explored for parameterizing a physically based deformable model, and how we measure the deformation.

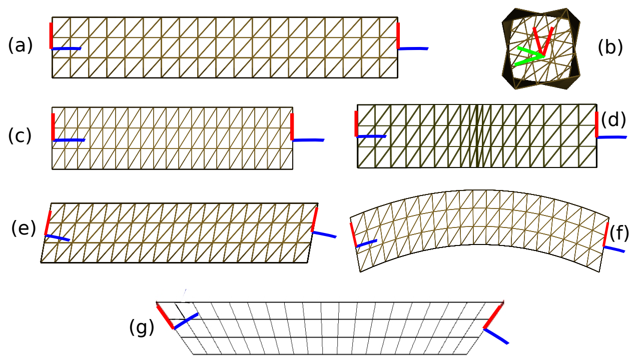


Fig. 1.2: Deformation modes obtained using two rigid frames. (a): rest shape, (b): twisting, (c),(d): compression with linear (resp. nonlinear) shape functions, (e): shear, (f): bending can be obtained using skinning, but (g): not using GMLS.

Linear blend skinning: The simplest and most popular blending method is *linear blend skinning* [24] where the displacements of control nodes q_i are locally combined according to their shape function w_i. 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:

$\mathbf{u} = \sum_i w_i(\bar{\mathbf{p}})\mathbf{A}_i\bar{\mathbf{p}}^* - \bar{\mathbf{p}}$, where $(\bar{\mathbf{p}})^*$ denotes a vector of polynomials of dimension d in the coordinates of $\bar{\mathbf{p}}$. 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 $3\times d$ matrix $\mathbf{A}_i(q_i)$ represents the transformation of node i from its initial to its current position and is straightforwardly computed based on the independent DOFs q_i: 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. $w_i(\bar{\mathbf{p}})$ is the shape function of node i evaluated at $\bar{\mathbf{p}}$. In linear blend skinning, weights need to constitute a partition of unity ($\sum w_i(\bar{\mathbf{p}}) = 1$). 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 $w_j(\bar{\mathbf{x}}_i) = 0$, $\forall j \neq i$.

Dual quaternion skinning: Linear blend skinning suffers from well known volume loss artifacts when the relative displacement between nodes is large and non linear. To remedy this, extra nodes need to be inserted. Another solution, is to use a better blending function. For rigid frames, *dual quaternion blending* offers a good approximation of the linear interpolation of screws at a reasonable computational cost [20]. It provides a closed-form solution for more than two transforms contrary to screw interpolation that requires an iterative treatment. Here, the relative displacement of a rigid frame i is no more expressed using a 3×4 matrix $\mathbf{A}_i$, but using a 8d vector $\mathbf{a}_i = [\mathbf{a}_0^{iT}\,\mathbf{a}_\varepsilon^{iT}]^T$ where $\mathbf{a}_0^i$ (resp. $\mathbf{a}_\varepsilon^{iT}$) is a unit quaternion representing the rotation (resp. translation). Blended displacements are computed as normalized weighted sums of dual quaternions: $\mathbf{b}' = \sum w_i\mathbf{a}_i / \|\sum w_i\mathbf{a}_i\|$. Finally the blended dual quaternion is converted [20] into a 3×4 rigid transformation matrix $\mathbf{A}$ to transform material points: $\mathbf{u} = \mathbf{A}\bar{\mathbf{p}}^* - \bar{\mathbf{p}}$.

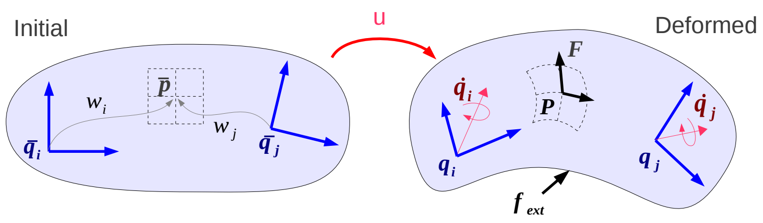


Fig. 1.3: The displacement u of a deformable object is discretized using nodes (blue). Strain is measured based on the deformation of local frames (black arrow) computed at integration points $\mathbf{p}$ in the material.

Strain Measure: As shown in Figure 1.3, the displacement is sampled at nodes, and is interpolated within the object based on nodal displacements. To apply the laws of continuum mechanics, we first need to measure the local deformation of the material. 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 $\mathbf{p}(\theta) = \bar{\mathbf{p}}(\theta) + \mathbf{u}(\theta)$. 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 called the deformation gradient $F = d\mathbf{p}/d\theta$, with reference value $\bar{F} = d\bar{\mathbf{p}}/d\theta$, 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 measures have been proposed, but all of them fit in our framework. For instance, the popular Green-Lagrange strain tensor, which is well suited for large displacements is computed as $(F^T\bar{F}^{-T}\bar{F}^{-1}F - \mathbf{I})/2$. Its six independent terms can be compactly stored in a $6d$ vector: $\varepsilon(\theta) = [\varepsilon_{xx}\,\varepsilon_{yy}\,\varepsilon_{zz}\,\varepsilon_{xy}\,\varepsilon_{yz}\,\varepsilon_{zx}]^T$.

1.4 The dynamics of frame-based continuum

This section explains how to set up the classical differential equation of dynamics for our models. An overview of the algorithm is given at the end of the Section. As shown in the following diagram, we apply a classical hyperelastic scheme: from the degrees of freedom $\mathbf{q}$, we interpolate a displacement field based on skinning, from which we compute the strain through spatial differentiation (Section 1.3). The elastic response $\sigma(\varepsilon)$ generates the elastic forces, and is a physical characteristic of the material. 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 (Section 1.4.4): $\mathscr{W} = \int_{\mathscr{V}} \int_0^{\varepsilon} \sigma^T d\varepsilon$. The associated elastic forces $\mathbf{f}$ are computed by differentiating the energy with respect to the DOFs (Section 1.4.1). After time integration (Section 1.4.3), we obtain the acceleration, velocity and the new position of each node.

$$\begin{tikzcd}
\text{Position} & \text{Strain} & \text{Energy} \\
\mathbf{q} \arrow[r, "\partial"] & \varepsilon \arrow[r, "\text{Material}"] & \mathscr{W} \arrow[d, "\partial"] \\
\dot{\mathbf{q}} \arrow[u, "\int"] & \ddot{\mathbf{q}} \arrow[l, "\int"'] & \mathbf{f} \arrow[l, "\text{Mass}"'] \\
\text{Velocity} & \text{Acceleration} & \text{Force}
\end{tikzcd}$$

1.4.1 Elastic force

The associated elastic forces **f** are computed by differentiating the energy with respect to the DOFs. Here, we explicitly introduce the deformation gradient F in the force computation:

$$\mathbf{f} = -\frac{\partial \mathscr{W}}{\partial \mathbf{q}}^T = -\int_{\mathscr{V}} \frac{\partial \varepsilon}{\partial \mathbf{q}}^T \sigma = -\int_{\mathscr{V}} (\frac{\partial \varepsilon}{\partial F}\frac{\partial F}{\partial \mathbf{q}})^T \sigma \tag{1.1}$$

This provides us with great modularity: the material module computes $\sigma(\varepsilon)$, the strain module computes $\partial\varepsilon/\partial F$, while the interpolation module computes $\partial F/\partial \mathbf{q}$, and the three can be designed and reused independently. This modularity allows us to implement the blending of rigid, affine and quadratic primitives using diffent techniques (e.g., linear blend skinning, dual quaternion skinning), 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.

One additional differentiation provides us with the stiffness $\partial \mathbf{f}/\partial \mathbf{q}$, 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:

$$\begin{aligned}\delta(\mathbf{f}) &= -\int_{\mathscr{V}} \frac{\partial \varepsilon}{\partial \mathbf{q}}^T \frac{\partial \sigma}{\partial \varepsilon}\frac{\partial \varepsilon}{\partial \mathbf{q}}\delta(\mathbf{q}) &&- \int_{\mathscr{V}} \delta(\frac{\partial \varepsilon}{\partial \mathbf{q}})^T \sigma \\ &= -\int_{\mathscr{V}} (\frac{\partial \varepsilon}{\partial F}\frac{\partial F}{\partial \mathbf{q}})^T \frac{\partial \sigma}{\partial \varepsilon}(\frac{\partial \varepsilon}{\partial F}\frac{\partial F}{\partial \mathbf{q}})\delta(\mathbf{q}) &&- \int_{\mathscr{V}} \left(\delta(\frac{\partial \varepsilon}{\partial F})\frac{\partial F}{\partial \mathbf{q}} + \frac{\partial \varepsilon}{\partial F}\delta(\frac{\partial F}{\partial \mathbf{q}})\right)^T \sigma\end{aligned} \tag{1.2}$$

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.

1.4.2 Visual and contact surfaces

Visual and contact surfaces can be attached to the deformable objects using the skinning method presented in section 1.3. Our framework sets no restriction on the collision detection and response methods. Any force $\mathbf{f}_{ext}$ 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:

$$\mathbf{f}+ = \frac{\partial \mathbf{p}}{\partial \mathbf{q}}^T \mathbf{f}_{ext} \tag{1.3}$$

1.4.3 Differential equation

Lagrangian mechanical models obey the following ordinary differential equation in generalized coordinates:

$$\mathbf{M}\ddot{\mathbf{q}} - \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{f}_{ext}(\mathbf{q}, \dot{\mathbf{q}}) \tag{1.4}$$

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 (elastic) forces, $\mathbf{f}_{ext}$ the external and inertial forces. Without loss of generality we consider Implicit Euler integration (see *e.g.*, [5]), which computes velocity updates by solving the following equation:

$$\left(\mathbf{M} - h\mathbf{C} - h^2\mathbf{K}\right)\delta\dot{\mathbf{q}} = h\left(\mathbf{f}_{ext} + h\mathbf{K}\dot{\mathbf{q}}\right) \tag{1.5}$$

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 does not need 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 node i and j:

$$\mathbf{M}_{ij} = \int_{\mathscr{V}} \rho \frac{\partial \mathbf{p}}{\partial \mathbf{q}_i}^T \frac{\partial \mathbf{p}}{\partial \mathbf{q}_j}, \tag{1.6}$$

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. For affine and quadratic primitives, masses are constant and can be pre-computed based on the voxel grid. We also pre-compute the mass of rigid primitives and rotate them in run-time according to their current rotations.

1.4.4 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. The following table summarizes the degrees of the different quantities obtained with linear shape functions for different strain measures and primitives.

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 1.1: Polynomial degrees obtained with linear shape functions and linear blend skinning.

Classical cubature methods 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 [4], 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 $\mathscr{V}e$ centered on $\bar{\mathbf{p}}$, we express points as $\bar{\mathbf{p}} + \delta(\bar{\mathbf{p}})$. The integration of order n of a scalar quantity v in this region can be written as $\int_{\mathscr{V}_e} v = \mathbf{v}^T \int_{\mathscr{V}_e} \delta(\bar{\mathbf{p}})^n$ where $\mathbf{v}$ is a vector containing the quantity v and its spatial derivatives up to degree n, and $\int_{\mathscr{V}_e} \delta(\bar{\mathbf{p}})^n$ is the integrated polynomial basis of order n over the region. The last term can be accurately estimated at initialization time using a voxel grid, as the one shown in Figure 1.4. This integration is exact if n is the polynomial degree of v. This formulation generalizes the concept of elastons [25] where quantities of order $n = 2$ are explicitly integrated in cuboid regions. Here, we consider arbitrary regions, and orders. Note that, using $n = 0$, the integration scheme is the classical midpoint rule: $\int_{\mathscr{V}_e} v \approx v\mathscr{V}e$.

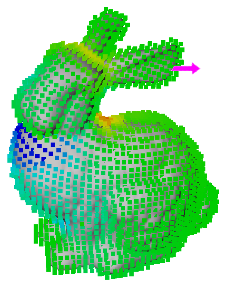


Fig. 1.4: In this example 15000 integration samples are generated by rasterizing a bunny model, and a midpoint (zero order) integration scheme is used. Colors represent a hue mapping of the strain.

The method presented in section 1.5 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 $\mathscr{V}e$ 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.

```
Data: Voxel map of material properties, number of control nodes
Initialization:
•  Distribute the control nodes // sec. 1.5.4
•  Compute the shape functions // sec. 1.5.3
•  Compute the mass matrix // sec. 1.4.3
•  Generate the integration samples // sec. 1.4.4
•  Compute the weights of the surface vertices // sec. 1.4.2

Loop:
•  Accumulate force from each integration point: // eq. 1.1
     –  Compute $F$ (and its spatial derivatives);
     –  Compute $\varepsilon$ (and derivatives) from $F$ using a given strain measure;
     –  Compute $\sigma$ (and derivatives) from $\varepsilon$ using a given material model;
     –  Add integrated force to each influencing primitive;
•  External forces and collision handling // eq. 1.3
•  At each solver iteration: // eq. 1.5
     –  Accumulate force change from each integration point // eq. 1.2
```

Algorithm 1: Deformable model computations.

1.5 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 1.1(d). In this section, we propose a method to automatically compute such functions.

1.5.1 Compliance distance

Consider the deformation of a heterogeneous bar in one dimension, as shown in Figure 1.5, 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 1.5(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

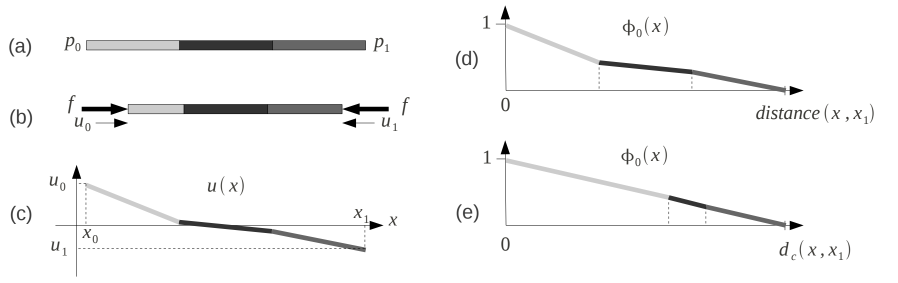


Fig. 1.5: Shape function based on compliance distance. (a): A bar made of 3 different materials, in rest state, with stiffness proportional to darkness. (b): The bar compressed by an external force. (c): The displacement across the bar. (d): The ideal w_0 shape function to encode the material stiffness. (e): The same, as a function of the compliance distance.

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 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 $\varepsilon = \frac{d\mathbf{u}}{dx} = \mathbf{f}/E = \mathbf{f}c$, 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 1.5(d) is exactly:

$$w_0(x) = \frac{\mathbf{u}(x) - \mathbf{u}(x_1)}{\mathbf{u}(x_0) - \mathbf{u}(x_1)} = \frac{\int_{x}^{x_1} c\, dx}{\int_{x_0}^{x_1} c\, dx} \tag{1.7}$$

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 1.5(e), and it can be computed without solving an equation.

1.5.2 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 [32], 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 1.7. 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.

1.5.3 Voronoi kernel functions

In meshless frameworks, each node is associated with a kernel function which defines its influence in space, as presented in Section 1.3. 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 1.1d, 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 [36]. 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

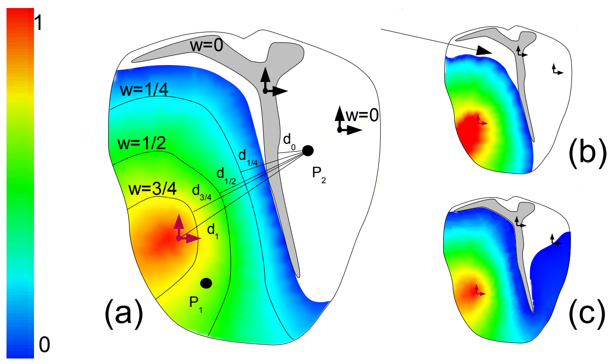


Fig. 1.6: Color map of the normalized shape function corresponding to the red node, computed using two Voronoi subdivisions (left), one subdivision (top right) and five subdivisions (bottom right).

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 1.6a 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 1.10d. 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. 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 1.6b, 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 1.6c, 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.

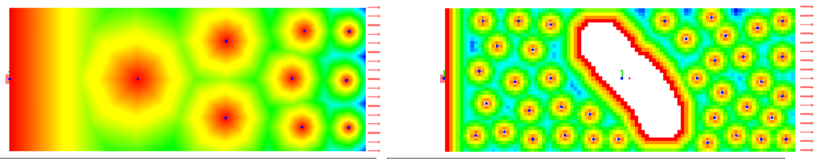
1.5.4 Node distribution

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 [1, 25]. 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 [20], at the price of more complex mechanical computations due to normalization.

1.6 Results

1.6.1 Validation

We implemented our method within the SOFA framework [3] to exploit its implicit and static solvers, as well as its GPU collision detection and response [2]. 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



Can't show picture #1 because `main-pics.pdf` not found. Create it from `main.dvi` using `dvips` and `ps2pdf`!

Can't show picture #2 because `main-pics.pdf` not found. Create it from `main.dvi` using `dvips` and `ps2pdf`!

Fig. 1.7: 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

shown in Figure 1.7. 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, as

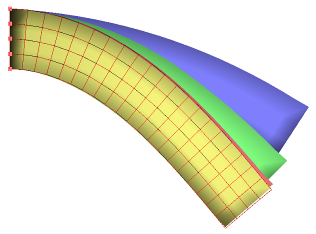


Fig. 1.8: Comparison of our models (solid colors) with FEM (wireframe). Blue: with two affine frames. Green: three affine frames. Red: five affine frames. Yellow: nine affine frames

illustrated in Fig. 1.8. We used regularly spaced nodes along the axis, with piecewise linear weight functions. We apply an extension force to the beam and verify that the force-extension law precisely matches the theoretical St. Venant-Kirchhoff model $f = \varepsilon + 3\varepsilon^2/2 + \varepsilon/2$, independently of the number of frames and the volume sample densities. Bending is more complex because it simultaneously involves extension-compression and shear, especially with large displacements as shown in the example in Figure 1.8. This confirms that accurate continuum mechanics can be performed using our model. The behaviors converge as we increase the number of nodes. As usual, fewer degrees of freedom result in more stiffness.

1.6.2 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 1.7, 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 1.2 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 [23] would speed up the precomputations in fine grids.

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 1.2: Timings.

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. 1.2). 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 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 1.2). In our implementation, linear blend skinning of rigid frames is about five times faster than dual quaternion skinning [13] 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.

1.6.3 Simulations

The most appealing feature of our method is probably its ability to easily model deformable objects using a reduced number of control nodes. The T-shaped rubber object shown in Figure 1.9 (Young's modulus $E = 200kPa$, Poisson's ratio $\nu = 0.3$) exhibits compression, shear, bending and torsion using only two frames, corresponding to a total of 12 DOF. The same number of DOF only allows to model a single linear tetrahedron in FEM, which can not exhibit torsion and bending! The object automatically exhibits an asymmetric stiffness reflecting its asymmetric shape.

Figure 1.11 shows a close-up of the high speed simulation presented in Figure 1.10, 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 [32] 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 1.12. In this model, 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

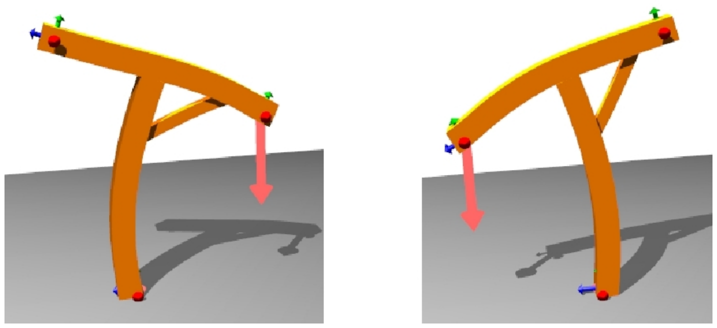


Fig. 1.9: An object with asymmetric stiffness automatically computed based on its asymmetric shape.

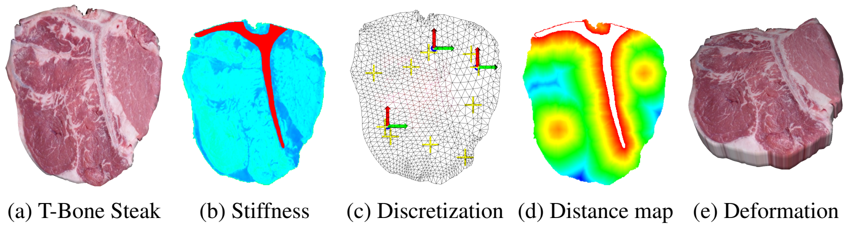


Fig. 1.10: The T-bone steak (a) has a rigid bone and softer muscle and fat, as seen in the volumetric stiffness map (b). Our method can simulate it using only three moving frames and ten integration points (c), running at 500 Hz on an ordinary PC. The frame placement is automatically generated using 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.

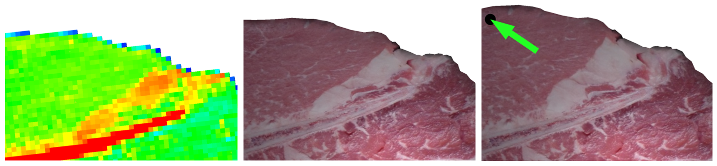


Fig. 1.11: The flesh and the fat, although interpolated between the same two control frames, pulled at the black point, exhibit different strains due to different stiffnesses.

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.

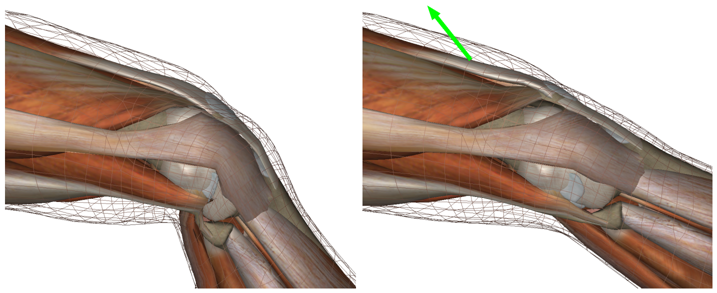


Fig. 1.12: Interactive knee simulation using 10 nodes. Pulling the quadriceps lifts the tibia.

Adding mechanical degrees of freedom during the simulation by inserting new frames with custom radial-basis shape functions is dramatically simpler than editing the mesh of an FEM model. In Figure 1.13, we show that a dynamically inserted frame at the contact point with an object can be used to generate a local deformation. The range of the local deformation can be tuned using the shape function of the inserted frame. Such a high level of adaptivity in a physical model is straightforward with our model, while it is difficult to implement using previous methods.

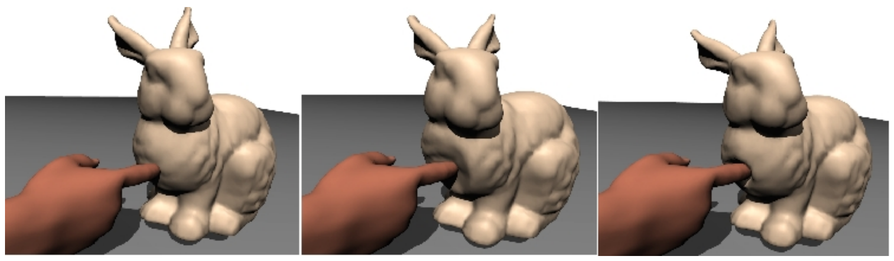


Fig. 1.13: More or less global deformation produced by dynamically inserting a frame with two different weight functions.

Our method can also be used to easily simulate physically-based secondary motions from skeleton-based animation or motion capture, as illustrated in Figure 1.14. 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.

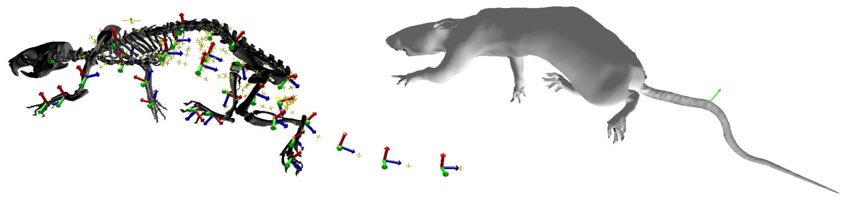


Fig. 1.14: Mix of animation and simulation. Left: model subset animated using motion capture and added physical nodes. Right: result, with flesh and tail animated using physics.

1.7 Conclusion

In this chapter, we have presented a new type of deformable model using continuum mechanics applied to objects undergoing skinning deformation fields. Our approach allows the creation of sparse meshless models with arbitrary constitutive laws, and we have demonstrated it using St. Venant-Kirchhoff materials. Moreover, 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. We have shown that the behavior of heterogeneous objects with complex materials and geometries can be simulated using a small number of control nodes and small computation times. The models are robust to large displacements and deformations.

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. Sampling is easier than with traditional particle-based meshless methods because there is no constraint on the number and on the placement of the nodes. Compared with FEM, adaptivity is easier because no volumetric mesh is used. However, due to computational time issues, the shape functions of the dynamically inserted nodes are currently limited to analytic radial-basis functions with local support. A faster computation of material-aware shape functions and hardware implementations are currently under investigation.

Acknowledgements

We would like to thank Florent Falipou, Michaël Adam, Laurence Boissieux, François Jourdes, Estelle Duveau and Lionel Revéret for models and 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

1. B. Adams, M. Ovsjanikov, M. Wand, H-P. Seidel, and L.J. Guibas. Meshless modeling of deformable shapes and their motion. In *Symposium on Computer Animation*, pages 77–86, 2008.
2. J. Allard, F. Faure, H. Courtecuisse, F. Falipou, C. Duriez, and P. Kry. Volume contact constraints at arbitrary resolution. *ACM Transactions on Graphics*, 29(3), 2010.
3. Jérémie Allard, Stéphane Cotin, François Faure, Pierre-Jean Bensoussan, François Poyer, Christian Duriez, Hervé Delingette, and Laurent Grisoni. SOFA - an open source framework for medical simulation. In *Medicine Meets Virtual Reality, MMVR 15, 2007*, pages 1–6, Long Beach, California, Etats-Unis, 2007.
4. Steven S. An, Theodore Kim, and Doug L. James. Optimizing cubature for efficient integration of subspace deformations. *ACM Trans. Graph.*, 27(5):1–10, 2008.
5. D. Baraff and A. Witkin. Large steps in cloth simulation. *SIGGRAPH Comput. Graph.*, 32:106–117, 1998.
6. Jernej Barbič and Doug L. James. Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Transactions on Graphics (SIGGRAPH 2005)*, 24(3):982–990, August 2005.
7. K. Bathe. *Finite Element Procedures*. Prentice Hall, 1996.
8. S. Cotin, H. Delingette, and N. Ayache. Real-time elastic deformations of soft tissues for surgery simulation. *IEEE TVCG*, 5:62–73, 1999.
9. G. Debunne, M. Desbrun, M-P. Cani, and A. Barr. Dynamic real-time deformations using space and time adaptive sampling. *SIGGRAPH Comput. Graph.*, pages 31–36, 2001.
10. F. Faure, B. Gilles, G. Bousquet, and D.K. Pai. Sparse meshless models of complex deformable solids. *ACM Transactions on Graphics*, 30(4), 2011.
11. T-P. Fries and H.G. Matthies. Classification and overview of meshfree methods. Technical report, TU Brunswick, Germany, 2003.
12. Nico Galoppo, Miguel A. Otaduy, Will Moss, Jason Sewall, Sean Curtis, and Ming C. Lin. Controlling deformable material with dynamic morph targets. In *Proc. of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, feb 2009.
13. B. Gilles, G. Bousquet, F. Faure, and D.K. Pai. Frame-based elastic models. *ACM Transactions on Graphics*, 30(2):15, 2011.
14. J.-P. Gourret, N. M. Thalmann, and D. Thalmann. Simulation of object and human skin formations in a grasping task. *SIGGRAPH Comput. Graph.*, 23(3):21–30, 1989.
15. E. Grinspun, P. Krysl, and P. Schröder. Charms: A simple framework for adaptive simulation. *SIGGRAPH Comput. Graph.*, pages 281–290, 2002.
16. M. Gross and H. Pfister. *Point-Based Graphics*. Morgan Kaufmann, 2007.
17. G. Irving, J. Teran, and R. Fedkiw. Tetrahedral and hexahedral invertible finite elements. *Graph. Models*, 68(2), 2006.
18. Doug L. James and Dinesh K. Pai. Multiresolution green's function methods for interactive simulation of large-scale elastostatic objects. *ACM Trans. Graph.*, 22:47–82, January 2003.
19. Peter Kaufmann, Sebastian Martin, Mario Botsch, and Markus Gross. Flexible simulation of deformable models using discontinuous galerkin fem. In *Symposium on Computer Animation*, 2008.
20. Kavan, Collins, Zara, and O'Sullivan. Skinning with dual quaternions. In *Symposium on Interactive 3D graphics and games*, pages 39–46, 2007.

21. Ladislav Kavan, Steven Collins, Jiri Zara, and Carol O'Sullivan. Geometric skinning with approximate dual quaternion blending. volume 27. ACM Press, 2008.
22. Theodore Kim and Doug L. James. Skipping steps in deformable simulation with online model reduction. *ACM Trans. Graph.*, 28:123:1–123:9, December 2009.
23. Yang Liu, Wenping Wang, Bruno Lévy, Feng Sun, Dong-Ming Yan, Lin Lu, and Chenglei Yang. On Centroidal Voronoi TessellationEnergy Smoothness and Fast Computation. *ACM Trans. Graph.*, 28, 08 2009.
24. N. Magnenat-Thalmann, R. Laperrière, and D. Thalmann. Joint dependent local deformations for hand animation and object grasping. In *Graphics interface*, pages 26–33, 1988.
25. S. Martin, P. Kaufmann, M. Botsch, E. Grinspun, and M. Gross. Unified simulation of elastic rods, shells, and solids. *SIGGRAPH Comput. Graph.*, 29(3), 2010.
26. Sebastian Martin, Peter Kaufmann, Mario Botsch, Martin Wicke, and Markus Gross. Polyhedral finite elements using harmonic basis functions. *Comput. Graph. Forum*, 27(5), 2008.
27. M. Müller, B. Heidelberger, M. Teschner, and M. Gross. Meshless deformations based on shape matching. *ACM Transactions on Graphics*, 24(3):471–478, 2005.
28. M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *Symposium on Computer Animation*, pages 141–151, 2004.
29. Matthias Müller and Markus Gross. Interactive virtual materials. In *Graphics Interface*, 2004.
30. B. Nadler and M.B. Rubin. 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, 2003.
31. A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson. Physically based deformable models in computer graphics. In *Comput. Graph. Forum*, volume 25 (4), pages 809–836, 2005.
32. M. Nesme, P. Kry, L. Jerabkova, and F. Faure. Preserving Topology and Elasticity for Embedded Deformable Models. *SIGGRAPH Comput. Graph.*, 2009.
33. J. O'Brien and J. Hodgins. Graphical models and animation of brittle fracture. *SIGGRAPH Comput. Graph.*, pages 137–146, 1999.
34. Dinesh K. Pai. STRANDS: Interactive simulation of thin solids using Cosserat models. *Computer Graphics Forum, The International Journal of the Eurographics Association*, 21(3):347–352, 2002.
35. S.M. Platt and N.I. Badler. Animating facial expressions. *SIGGRAPH Comput. Graph.*, pages 245–252, 1981.
36. M. J. D. Powell. The theory of radial basis function approximation. *University Numerical Analysis Report*, 1990.
37. Eftychios Sifakis, Kevin G. Der, and Ronald Fedkiw. Arbitrary cutting of deformable tetrahedralized objects. In *Symposium on Computer Animation*, 2007.
38. D. Terzopoulos and H. Qin. Dynamic nurbs with geometric constraints for interactive sculpting. *ACM Trans. Graph.*, 13(2):103–136, 1994.
39. Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In Maureen C. Stone, editor, *SIGGRAPH Comput. Graph.*, volume 21, pages 205–214, July 1987.
40. Matthias Teschner, Bruno Heidelberger, Matthias Muller, and Markus Gross. A versatile and robust model for geometrically complex deformable solids. In *CGI*, 2004.
41. J.A. Weiss, B.N Maker, and S. Govindjee. Finite element implementation of incompressible, transversely isotropic hyperelasticity. *Comp. Meth. in App. Mech. and Eng.*, 135:107–128, 1996.