ACCURATE INTERACTIVE ANIMATION OF DEFORMABLE MODELS AT ARBITRARY RESOLUTION

Matthieu NESME, Francois FAURE

LJK Laboratory, CNRS-INRIA, Grenoble University, FRANCE
matthieu.nesme@gmail.com, Francois.Faure@imag.fr

Yohan PAYAN

TIMC-IMAG Laboratory, CNRS-UJF, Grenoble University, FRANCE
Yohan.Payan@imag.fr

Providing realistic interactive simulation requires a powerful animation method with a highly detailed rendering. Based on continuum mechanics, the finite element method needs a volumetric representation of the object to animate. This paper proposes an automatic method for building meshes that are well adapted to interactive simulation starting from miscellaneous input data. Contrary to commonly used methods based on tetrahedral volume meshing, the object is embedded in a regular grid of deformable hexahedra at an arbitrary resolution. This alleviates the complexities and limitations of tetrahedra and results in regular, well-conditioned meshes. Mass and stiffness are set in order to model the physical properties as accurately as possible at any given resolution, in a manner that takes into account the distribution of material within the hexahedra. This allows us to accurately model the mechanical properties of the partially empty boundary hexahedra, and thus enables us to perform fast simulation at a coarse resolution. The accuracy of this approach is compared to theoretical results. In addition, we extend a fast and robust co-rotational approach to the case of hexahedral elements. This permits simulation of arbitrarily complex shapes at interactive rates. We show how to build the hexahedra directly from surfaces and from segmented scanned data, which is very useful to animate complex artistic models or patient specific models for individual medical simulation. Finally, we show how a fast volumetric rendering can make efficient use of the grid structure.

Keywords: interactive simulation, soft bodies, physically based animation, finite elements, surgical simulator, patient-specific meshing

Introduction

Soft body simulation is a growing research domain for several communities, such as Computer Aided Surgery and Computer Graphics. Indeed, the tissues and organs of a patient are deformable objects in much the same way as the clothes or skin of an animated character. Even if the interest of simulating such objects is quite different depending on the final application (whether a surgical simulation or a video game), the mechanics which govern them are the same. In what follows, we describe in more detail why these communities have the same need for interactive deformable models through their various objectives.

Computer aided surgery (CAS) aims at assisting surgeons for the realization of diagnostic and therapeutic gestures in a rational and quantitative way in order to increase safety and accuracy. While early systems focused on orthopaedics, more recently research has turned to modelling anatomical structures mainly composed of biological soft tissues. The corresponding CAS systems therefore need to take into account the deformations of such structures. In most cases, authors propose to build biomechanical models of the anatomical structures and use these models to predict, in the most accurate way, the tissue deformations induced by the surgical gesture. Finding a solution often requires a slow static computation. The community is now looking for models that could be used intra-operatively, with real-time re-planning of surgical gestures. In parallel, surgical simulation systems have been provided. These surgical systems could be a great help in the learning and training processes, allowing the surgeon to acquire, for example, hand-eye coordination through repetition of the most difficult gestures, or to choose the best surgical procedure for a given pathological case. In such systems, perfect accuracy is not necessary, a *physical plausibility* could be sufficient.

The Computer Graphics community has developed methods for visually plausible dynamical animation of complex physical soft objects. Considerable speedups have been obtained. It must now focus on the accuracy of the deformations, in order to be as realistic as possible, in comparison with real data. Some recent works try to provide mechanical models with innovative implementations that preserve a continuous modelling context while proposing robustness and fast computational time.

The two communities have the same needs for soft body modelling: accuracy, robustness and interactivity (*i.e.* fast computation). However, physically based animation of soft objects is not yet widely used in real-time applications because physical models are difficult to tune (like trade-off speed/robustness parameters) and their animation remains computationally expensive. In order to obtain satisfying results, continuum mechanics must be employed to compute forces. The most common method employed to discretize equations is the Finite Element Method, that needs a volumetric representation of the objects. Unfortunately, building such meshes is a difficult task, particularly for very complex models designed by an artist, or for patient specific mesh in medical simulation.

In order to obtain a good trade-off between accuracy and speed, it is very important to control easily the number of mechanical degrees of freedom (DOF). However, control of the

mechanical mesh resolution is not easy. Indeed, it is extremely challenging to automatically build a Finite Element mesh for an object at a given resolution. This point is very important in the CAS context, where simulations need to be adapted to each patient's anatomy. Currently, each case is treated as a specific case and it is desirable to automate the processing. Moreover, in the case of interactive simulations, it is important to obtain regular and well-shaped meshes that give well-conditioned systems which are faster to solve.

Another point, which should be emphasized, is the quality of rendering. Using most classical animation approaches, it is complicated to obtain a highly detailed rendering with interactive mechanics, because both resolutions are identical. In classical meshing, the resolution of the volumetric mesh used for mechanics is based on the surface mesh used for rendering. Therefore, one must choose between a good rendering with slow mechanics or fast mechanics with a poor rendering. Here, we present an approach which decouples the rendering and mechanical resolutions. This permits fast animation of complex objects through the use of a highly detailed model and a coarse representation of the mechanics.

In this paper, a novel approach is proposed to start to tackle these difficulties in the case of viscoelastic deformable bodies. Our approach can be summarized as follows. We first start from volumetric data scan, or build a high-resolution voxelization of a geometrical surface. Voxels are then recursively merged up to an arbitrarily coarser mechanical resolution. The merged voxels are used as hexahedral finite elements embedding the detailed geometrical shape, and are animated using fast implicit time integration. At each level, the mass and stiffness of a merged voxel are deduced from its eight children, automatically taking into account the non-uniform distribution of material. This technique handles objects which simultaneously include volumetric parts, surface parts and one-dimensional parts. Moreover, the animation is robust against degenerate configurations such as element inversion. These features make our approach ideal for interactive virtual environments such as medical simulators. Automatic meshing directly from volumetric data permits the easy use of patient specific models or the instantaneous animation of any model created by an artist. A mechanical resolution suitable for an interactive simulation can be easily chosen while keeping a physical plausibility thanks to the finite element formulation based on a non-uniform distribution of material. In the animation of scanned data, a fast volumetric rendering can make efficient use of the mechanical hexahedral structure.

Our specific contributions include: the automatic voxelization from segmented scanner data or from surface objects, the automatic tuning of the FEM parameters based on the distribution of material in each hexahedron, the robust animation of hexahedral elements and the rendering of the volumetric data using the same data structure as the mechanics. Moreover, the accuracy of our modeling framework is compared with certified standards.

The remainder of this paper is organized as follows. Related work is briefly discussed; we show how to construct the hexahedral mesh and embed the surface in the deformable hexahedra; a new method to apply fast dynamic hexahedral FEMs is presented and how to interact with the embedded surface is detailed; how to tune the physical properties of the hexahedra and evaluates their accuracy is explained; we present how to render efficiently

4   *Nesme M., Faure F., Payan Y.*

deformed volumetric data; results are discussed and a conclusion is given.

Related Work

Building a patient specific Finite Element (FE) mesh is usually a delicate and time-consuming task. From the results of Computed Tomography or Magnetic Resonance Imaging, volumetric data can be extracted through image segmentation, providing the external contour of an organ as well as its intrinsic sub-structures. Examples of original and segmented images of a CT-scan are presented in Figure 1. From this voxel grid, a polygonal surface is extracted with a Marching Cubes-like algorithm [1]. Using this geometrical data, 3D FE meshes are built and used to discretize the mechanical formulation. It can be challenging to mesh the interior of such models with a controllable number of elements especially with hexahedra. Moreover, tetrahedra computations are faster and for this reason more common in interactive simulations. However, tetrahedral meshing has several significant disavantages. Tetrahedra meshing can create nearly singular elements which result in unstable simulations, especially for small parts. Secondly, tetrahedral meshes with linear shape functions tend to lock for incompressible material (*i.e.,* Poisson's ratio close to 0.5) unless special countermeasures are applied [2]. Thirdly, most artist designed models are not adapted for volumetric meshing. Highly detailed and curvature-dependant models often comprise very fine surfaces and even lines, that are impossible to animate using a classical tetrahedral mesh. Examples of complex artist designed models are presented in Figure 1. Note the holes (dragon model), thin surfaces (dragon wings), very small details (dragon's teeth) and the quasi-linear parts (mouse's tail). Arguing against the time consuming component of this specific FE object or patient-mesh elaboration, others have proposed using a reference FE mesh (an "atlas", built once from a reference object geometry) and matching this mesh to each new patient morphology [3–5]. These methods reduce the time needed to build the patient-specific FE models, but they are still very specific and may alter the regularity and/or the quality of the elements during the registration phase. Since initial data is already volumetric, it is possible to work directly from the raw data. In biomechanics in the 90's, Keyak and colleagues introduced the idea of building patient-specific meshes by using each voxel of a CT-scan as a hexahedral finite element [6]. With this approach, data is not transformed into an approximated surface representation and then back to a volumetric representation. This is advantageous, especially when intermediate algorithms are complex and time-consuming, give only approximations and can produce degenerated elements. A great advantage of this method is that no geometric details are lost lost from the scanner information and CT gray levels are used for element stiffness values. On the other hand, the resulting meshes are enormous and suffer from unreasonable runtimes, they are impossible to animate interactively. To increase element size and decrease mesh size, one approach is to embed the entire object in a base grid at the desired resolution (using either a Cartesian grid or an octree), to remove elements lying outside the part, and to project external nodes onto the surface [7]. This proves to be an elegant solution when the surface is quite smooth, like in the case of bony structures [8]. However, element sizes have to be

much smaller than the object itself, otherwise, the projection can generate bad elements for small parts and results in a mesh with poor quality near the boundary. On the other hand, as already stated, too many small elements cannot be animated interactively. Note that there are recent methods that try to improve element quality, such as [9].

The complexity of the material laws has a high impact on computation time. Viscoelasticity can be used to model a wide range of real-world objects, including biological tissues. They are modeled as a continuous material subject to physical laws relating local strain to local stress, and discretized over a finite number of degrees of freedom to allow numerical simulation. Such models have been studied in the domain of mechanical engineering [10], and a wide variety of material laws and discretization methods have been proposed. In the Computer Graphics community, the seminal paper of Terzopoulos [11] showed how to produce complex and visually realistic animations, at a high computational price. Since then, a lot of work has been done to reduce the complexity using alternative material laws, spatial discretization, and time integration. In order to improve the computational efficiency of continuous biomechanical models, researchers have proposed several new approaches. The most simple viscoelastic law is the spring model [12, 13]. However, it has been shown that it can not accurately model 3D elasticity, and recent work focuses on 3D finite elements [14]. Important gains in computation time have been obtained by replacing the complex Green-Lagrange rotationally invariant strain tensor by the product of a rotation with the linearized Cauchy strain tensor [15–17]. This method is commonly referred as *co-rotational*. Robustness to degenerate configurations, such as flat or inverted elements, has since been improved [18, 19].

Spatial discretization directly affects the number of degrees of freedom, and thus, the complexity of the system. This is rarely applicable if one wants to maintain visually pleasing detailed shapes. Some approaches try to separate rendering detail from the (possibly hierarchical) mechanical model, using an external [20] or embedded [17, 21–23] rendering layer. Nevertheless, the tetrahedrization stage remains far from trivial. To avoid this volume meshing stage and to control the number and shape of the elements, some methods automatically build an optimized mechanical mesh using a 3D grid [24, 25] or an octree [26, 27]. However, the resulting mechanical properties are simplified and the meshes have to be very fine to model the objects accurately. Alternatively, it is possible to reduce the number of degrees of freedom of detailed shapes using modal analysis [28] or global shape matching [29], however these methods fail to capture local deformation. Recently, meshless methods have been proposed [30–32] in order to more easily model fracture and tearing, however they are quite slow and not adequate for real-time animation of non-pasty soft bodies.

Time integration is an important issue when dealing with stiff objects, since very small time steps are required to avoid instabilities when explicit schemes are used [33]. Baraff and Witkin [34] showed how to efficiently apply implicit time integration while allowing large time steps. Collision detection and response is also a difficult topic related to time integration. Many detection methods have been proposed [35] and response strategies have

6   *Nesme M., Faure F., Payan Y.*

been discussed [36].

Several volume rendering techniques have been proposed, the state of the art can be found in [37]. We follow the classical interactive approach, using 3d texture with an hardware-accelerated mapping [38,39]. The main idea is to cut the volume in several slices from back to front integrating the pixel intensity. These slices are simple, semi-transparent polygons to which the 3d texture is applied. Different methods exist to choose the orientation of the slices, some can introduce artifacts at certain viewing angle. Better quality is achieved when the slices are orthogonal to the viewing direction.

Our Approach

Our approach directly works on CT-scan data at an arbitrarily coarse resolution.

We sample the entire object using a grid as illustrated in 2D in Figure 2. Note that this is contrary to [7, 8] in which external nodes are projected. The advantage of this approach is that all details are keep and avoiding the use of projection prevents degenerate elements. Moreover, the element size can be easily controlled and all elements are regular and uniform, which results in a well conditioned system that is easier to solve. Contrary to the traditional finite elements, in our approach the volume mesh does not fit the object surface exactly (*i.e.,* all the external nodes are not on the surface). Indeed, some elements are at the object boundary, with nodes inside the object and other nodes outside the object (hatched elements in Figure 2). In the following, these hexahedra are called *bounding elements*. This idea has been used in computer graphics [17, 21, 22] and more recently in medical simulation [27, 40], but without considering the mechanical behavior of half empty hexahedra. In the following, we show how to correctly animate bounding elements by precomputing mechanical properties for each hexahedral element, taking into account the matter distribution and applying correct boundary conditions onto the surface.

As a first step, the three-dimensional hexahedral mesh is built directly from a set of slices provided by a medical scanner. It consists of a voxelization of the object at several resolutions. This starts from a three-dimensional set of two-dimensional slices and from the segmentation of an isolated organ in each slice as illustrated in figure 1. An octree representation is employed to condense the finest voxels given by initial data into a coarser mechanical resolution. Using the segmented data, voxels of the considered object can be easily identified as being outside, inside or along the boundary. Note that the boundary elements do not necessary need volumetric data to be built, but equivalent meshes can be built from implicit functions or surfaces like in [41] by using a voxelization. The animated object is linearly interpolated within the elements as illustrated in Figure 2-*right*. A great advantage of this approach is that surfaces, as well as lines (dragon's wings and mouse's tail in Figure 1), are handled in the same way as volumes. A nice feature applied to these kind of objects is that their rest shape can be straightforwardly chosen as the most familiar form such as the shape defined during the mesh processing.

It is also important to note that in most cases described in the literature, borders of the object are always approximated. Indeed, in the classic pipeline, the volumetric data are first approximated by a marching cube algorithm to build a surface. Then this surface has to be simplified further in order to obtain a coarser resolution mesh. Finally, to build a well conditioned volumetric mesh, the surface is often modified, like by using TetGen tetrahedral mesh generator [42]. With a classical mesh, even if the border seems precise, and boundary conditions are well applied, it remains a real approximation of the initial volumetric data. It is well known that obtaining an exact segmentation is a very difficult task, and sometimes impossible. Note that in such cases, with a simple predefinition of a zone of interest, our method can directly provide global deformations, even if behavior at the fuzzy borders will be inaccurate. This proves to be another advantage relative to classical meshing techniques.

Figure 3 compares results of a tetrahedrization with TetGen of the liver (shown in Figure 1) and meshes with our method at three different resolutions. The comparison is done with tetrahedra since they are the most popular elements in interactive simulation due to their computation speed. First, resulting tetrahedra are very numerous and do not have all the same size. Note that our mesh has fewer hexahedral elements for a similar resolution than the tetrahedral mesh, but uses many more nodes, since a node can only link a maximum of eight elements and at the border many nodes belong to only one element. With our mesh, forces are faster to compute, but the system is bigger so the time integration is slower, not forgetting that we have to update the surface interpolation. In this end, computation time is equal for a similar resolution, values are given in Figure 3. Results for the computation of dynamical tetrahedra deformations use [19] which employs a similar co-rotational calculation ; note that the computation of one hexahedron is slower than the computation of one tetrahedron.

Mechanical Animation

Once the collection of hexahedra is built, mechanical laws can be applied to bounding elements to animate the hexahedra and the interpolated shape as presented in [41].

Force Computation

The linear Finite Element Method is used to compute internal forces on hexahedral elements [10]. In this paper, we consider only isotropic linear elastic materials. According to Hooke's law, the material properties are defined by Young's modulus and Poisson's ratio. As explained in [15], a great advantage of the linear formulation is that all stiffness matrices can be precomputed, because they do not evolve significantly during the animation.

Since the standard linear approach is inaccurate given a large rotation of the elements, a co-rotational approach has to be employed to avoid artificial inflating. The main idea is to

compute the forces in an element's local rotated frame by decomposing the displacement into a rigid rotation combined with a deformation. The force applied by a deformed element to its sampling points is given then by

$$\mathbf{f} = \mathbf{R}^T \mathbf{K}(\mathbf{R}\mathbf{x} - \mathbf{x}^0)$$

where $\mathbf{K}$ is the stiffness matrix, $\mathbf{x}$ and $\mathbf{x}^0$ are the current and the initial positions, and matrix $\mathbf{R}$, which encodes the rotation of a local frame with respect to its initial orientation, is updated at each frame.

Since the first approach proposed [15], several methods have been proposed to compute $\mathbf{R}$. Methods using eigenvectors and eigenvalues [17,18,43] give the smallest deformations for most accurate results. A significantly faster method is proposed in [19]. Another important point of this approach is its robustness in degenerate configurations such as flat or inverted elements. An elegant treatment has also been presented in [18] but it is not aimed at real-time applications. Using [19], the inversion of an element is easily detected, and modeled as a high compression. Stability is maintained and the elements can recover their initial shape without visible artifacts. This is the essential and the desired effect in the case of non-physically plausible situations, which can occur in real applications.

We extend to hexahedra the method [19] initially designed for tetrahedra. For each hexahedron, three arbitrary edges could be selected in order to extract the rotation as is done for a tetrahedron. However, it is preferable to involve all the vertices in the computation to obtain a rotation that results in smaller measured deformations. To do this, the average of four edges in the three directions is computed.

Time integration

The approach used to compute forces is completely independent of the approach used to integrate over time. Nevertheless, the stable Euler implicit solver is employed to solve the ODE. The filtered conjugate gradient presented in [34] gives the advantage of not necessarily needing to assemble the stiffness matrix, since it requires only a matrix/vector product. It is well adapted to the co-rotational formulation for which the assembling matrix should vary at each time step because of rotations. Moreover, it is possible to process the elements one at a time. Implicit integration is very well-suited to our approach, where all elements are regular and are the same size. The corresponding equation system is well-conditioned and can be solved iteratively. As usual, a large number of iterations can be necessary to accurately model materials with high stiffness. However, a reduced number of iterations (between 5 and 10) is generally acceptable to obtain interactive animations. This allows for a trade-off between accuracy and computation speed.

Boundary Conditions

Applying classical boundary conditions to the element nodes is trivial but is not always sufficient in our method because certain elements are astride the surface. To directly manipulate the surface, or to accurately handle contact constraints, the boundary conditions must be applied to the surface, and dispatched over the lattice vertices.

Embedding the Object

The surface of the object is linked to the bounding elements: surface vertices $\mathbf{u}(p)$ are trilinearly interpolated using the eight values $\mathbf{u}(q)$ defined at the nodes of the bounding element as illustrated in 2D in Figure 2.

$$\mathbf{u}(p) = \mathbf{H}\mathbf{u}(q) \tag{1}$$

where the (3×24) interpolation matrix $\mathbf{H}$ of the considering element is the concatenation of the influences of the element vertices on a given vertex.

In order to be able to apply constraints on volumetric data where the surface is nonexistent, a virtual surface has to be created. For this, barycenters of boundary voxels at the scanner's resolution are considered; their positions are evaluated from the initial data and are used to sample the virtual surface.

Penalty forces

In case of penalty forces, forces $\mathbf{f}(p)$ are applied to points of the surface. We dispatch these forces over the lattice vertices. The virtual work principle implies that

$$\mathbf{f}(q) = \mathbf{H}^T \mathbf{f}(p)$$

where H is the interpolation matrix at point p (proof given in appendix). A result of penalty forces on a surface is illustrated in Figure 1 with the mouse under a plane.

Hard constraints

When a displacement (or a velocity) is imposed at the surface, the corresponding lattice displacements have to be computed. We compute the smallest displacements of the control points in least-squares sense, as in direct manipulation of FFD [44].
A displacement constraint is written $\mathbf{u}(p) = \mathbf{c} \Leftrightarrow \mathbf{H}\mathbf{u}(q) = \mathbf{c}$ where $\mathbf{c}$ the constraint value. When several constraints are applied, we build a system containing all constrained surface points $\mathbf{p}$ and all influenced control points $\mathbf{q}$. The resulting system can be solved using a pseudo-inverse of the assembled matrix $\mathbf{H}$:

$$\mathbf{u}(\mathbf{q}) = \mathbf{H}^+ \mathbf{c}$$

10   *Nesme M., Faure F., Payan Y.*

with $\begin{cases} \mathbf{H}^+ = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T \ , \ dim(\mathbf{p}) \leq dim(\mathbf{q}) \\ \mathbf{H}^+ = \mathbf{H}^T(\mathbf{H}\mathbf{H}^T)^{-1} \ , \ dim(\mathbf{p}) > dim(\mathbf{q}) \end{cases}$.

When there are more constraints than control points, the best compromise (in the least-square sense) is computed. Note that in the specific case of at most one constraint applied to each lattice vertex, $\mathbf{H}_i^T\mathbf{H}_i$ is a scalar, and the solution of the system is extremly efficient.

Accurate Physical Parameters

The main specificity of our method is the presence of bounding elements that are astride the surface. Therefore, these elements are "filled" to a varying degree. To adapt the mechanical behavior of this type of element according to its content, it is necessary to dispose of more precise meshes than that desired for the animation, where the mechanical resolution is coarser than the volumetric data resolution. The main idea is to project behavior properties, *i.e.,* stiffness and mass information, from fine voxels to coarser voxels.

Finest Resolution

First, all precomputations have to be done at the finest resolution using the classical method. Stiffness and mass matrices are computed for each of the fine hexahedral elements.

It is important to note here that parallel work is being done to identify material properties in each voxel of space using ultrasound or CT-scan [6]. This information would be very easy to take into account in the finest resolution and could be projected to the mechanical mesh.

Precomputed Non-Uniform Stiffness

Let us consider the child elements as interpolations of their parent element. In this context, it is possible to deduce the stiffness influence of a child element $\mathbf{K}_{child}$ on its parent $\mathbf{K}_{parent}$, and to deduce the stiffness of each parent based on its eight children. In this section, we take into account the distribution of the material over the child elements. Differences with the uniform stiffness are illustrated in Figure 4.

If only the large elements are considered, it corresponds to degrees of freedom from child nodes. To some extent child nodes are dependent on their parents and can be deduced as an interpolation (child nodes are constrained to stay "in the middle"). In this case, we can define eight matrices $\mathbf{L}_{child}$ which represent the interpolated child nodes $\mathbf{u}_{child}$ based on their parent nodes $\mathbf{u}_{parent}$: $\mathbf{u}_{child} = \mathbf{L}_{child}\mathbf{u}_{parent}$. Reciprocally, forces applied to child nodes can be popped up to the parent nodes using the transpose of the interpolation: $\mathbf{f}_{parent} = \mathbf{L}_{child}^T\mathbf{f}_{child}$. Moreover, $\mathbf{f}_{child} = \mathbf{K}_{child}\mathbf{u}_{child}$, so $\mathbf{f}_{parent} = \mathbf{L}_{child}^T\mathbf{K}_{child}\mathbf{L}_{child}\mathbf{u}_{parent}$. Summing

the influence of the eight children of a parent element, we obtain:

$$\mathbf{K}_{parent} = \sum_{i=0}^{7} \mathbf{L}_i^T \mathbf{K}_i \mathbf{L}_i$$

The same technique is used for masses: $\mathbf{M}_{parent} = \sum_{i=0}^{7} \mathbf{L}_i^T \mathbf{M}_i \mathbf{L}_i$, where $\mathbf{M}$ is the mass matrix of an element. At the mechanical level, we lump the mass matrix to obtain a diagonal matrix, which allows faster matrix products and easy weight computation.

Matrices $\mathbf{L}_{child}$ only depend on the shape of the elements, so they can be defined once for all. Matrices for hexahedral subdivision are given in Appendix.

Intuitive Results of Non-Uniform Hexahedra

Taking into account the contents of bounding elements improves animation quality without adding complexity. This allows us to perform plausible animations using a reduced number of elements. An extreme case is presented in Figure 5 where an object in form of 'c' is animated under gravity using one single boundary element. As expected, using an averaged stiffness, both parts of the object have the same properties, and the empty part is as stiff as the full part. In contrast, using our precomputed non-uniform law, stiffness takes into account where the matter is, resulting in a more realistic behavior.

Influence of the Hexahedra Positioning

The accuracy of the employed co-rotational hexahedral elements has already been tested in classical uses, where elements exactly fit the object. They give quite good results compared to theoretical results, certified numerical results and real data [45]. In this section, we want to evaluate the behavior of non-uniform elements. Remember that the bounding elements using non-uniform properties only appear at object borders. Although they do not represent a large percent of elements in pratice, they permit the simulation of realistic deformations at low cost.

Fixed Beam

The deviation of a fixed beam subject to gravity has already been measured for finite elements [45], and provides very satisfactory results.

Here, the accuracy of boundary elements is evaluated with elements that do not fit the beam, but on the contrary, that are laid out in the worst case (too large, rotated...) as presented in Figure 6. This configuration is not optimal but serves to test the accuracy of the non-uniform elements. Even if the voxel resolution is large in comparison with the simulated beam, results in best and non-optimal cases are almost similar. For a quantitative

comparison, Euclidean distances between all positions of nodes at equilibrium are computed between both rotated hexahedra and well fitted hexahedra (reference) to deduce the mean, min, max distances and standard deviation. The beam measures 160*40*40 mm^3. Results are presented in Figure 7.

Uni-axial Tension of a Cylinder

Whereas in the previous case a hexahedral domain is considered and can be easily meshed in a traditional way, it is more interesting to perform tests on a cylinder which is more complicated to mesh. In this case, bounding elements become of interest. Moreover, this example serves to evaluate the behavior of bounding elements that are partially filled.

An experiment is illustrated in Figure 8. For numerical simulations, two types of meshes are used, a classical one based on prisms and hexahedra, and one with bounding elements (presented in Figure 9). Solutions obtained with our method are compared with certified solutions using the Ansys software [46] which uses the non-linear Green-Lagrange strain tensor. Note that a mixed element mesh (hexahedra+prisms) is needed since the automatic Ansys tetrahedral mesh can not be used in Ansys because simulations do not converge for a quasi-incompressible material (Poisson's coefficient to 0.49), either because of non-symmetries or locking effects. Moreover, Ansys does not have automatic hexahedral meshing.

In order to test the influence of the bounding element positioning, two meshes have been employed. The first is centered around the cylinder whereas the second is shifted and thus not symmetrical. In both cases, although bounding elements do not fit the surface exactly, the cylinder circumference remains a circle under compression as presented for the median layer in figure 9. This remains true, even if the bounding box is shifted as in the last example. Results are coherent since under uni-axial compression, the plane of cut stays circular on a cylinder, as verified on Table 1, where cylinder radius are evaluated at each node. Since the Ansys mesh is built to be symmetrical, it is coherent that results are perfectly symmetrical.

Radius	Comp	Mean	Min	Max	Std Dev
Ansys	10%	0,5331052	0,5331052	0,5331052	0
	20%	0,5662103	0,5662103	0,5662103	0
	30%	0,5993155	0,5993155	0,5993155	0
Well Fitted Mesh	10%	0,5435298	0,5424220	0,5446057	0,0008812
	20%	0,5835664	0,5815420	0,5855633	0,0016283
	30%	0,6194658	0,6167130	0,6222278	0,0022409
Badly Fitted Mesh	10%	0,5429189	0,5412755	0,5445390	0,0009428
	20%	0,5828810	0,5796358	0,5859284	0,0018170
	30%	0,6189712	0,6141881	0,6232140	0,0026136

Table 1. Radius measurements for three compression rates.

Conclusion These two tests show that results are similar some is the placement. Bounding elements do not seem to have big accuracy drawback.

Volumetric Rendering

Since volumetric data can be directly animated, it could be very interesting to use it for rendering. The cell structure used for mechanics is very well adapted to volumetric rendering. Moreover, using the same data structure for animation and rendering saves memory and time for computation. In order to render the animated volumetric data, we apply a technique rather similar to [39]. This method computes intersections between slice and prisms in order to propose a complete hardware accelerated algorithm. In our approach, a new method is proposed for generating slices from back to front and for computing intersections between slices and deformed hexahedra to permit an interactive rendering of a deformable organ. To accelerate computational time, the extension to a hardware implementation would be possible in the same way as [39], and could be even more easily given the lastest graphic cards generation and the geometric shader.

The basic idea is to display semi-transparent planes through the entire object. Planes are oriented face to the viewing point in order to delete artifacts due to viewing angle. To handle semi-transparency using openGL, planes have to be rendered from back to front. In this optic, the first chosen plane is the one that passes through the furthest corner of the bounding box of the object. Others planes are chosen by moving towards the viewing point along the normal until the plane no longer intersect the organ. These planes are displayed progressively. Planes support the 3D texture and have to be triangulated to be rendered efficiency using openGL. To find triangles, all intersections between a plane and all edges of the deformed hexahedral mesh are considered, as illustrated in Figure 10 (the fast algorithm from [47] is employed). Note that in order to limit the number of intersection tests, each edge that has been intersected but that is no longer intersected, is exempted from

14   *Nesme M., Faure F., Payan Y.*

further tests. The triangulation is deduced by working cell by cell in order to return to an easier problem, searching the triangulation of a contour in 2D under the slice. Considering only small deformations that always give a convex polygon, a very fast triangulation can be employed. The idea is to separate vertices in four groups according to the sign of their cosines with two orthogonal lines. Deducing a triangle strip from a well-ordered list is a simple task as presented in appendix 1. To apply the 3D texture on these triangles, texture coordinates have to be determined. The basic idea is to save the texture coordinates, *i.e.,* the corresponding positions in the initial volumetric data, at each hexahedron corner during the mechanical mesh construction. These values do not vary during the animation. Then, texture coordinates at nodes of triangles can be evaluated as an interpolation from values at mesh nodes.

Quality strongly depends on the number of displayed planes, one hundred planes are sufficient to have a satisfying quality (in Figure 11 101 planes are used). Each time planes are added (or removed), in order to keep the same opacity, the transparency coefficient has to be modified following the formula: $alpha_{new} = alpha_{old} * nbplanes_{old}/nbplanes_{new}$.

Many interesting visualization tools can be added, for example, to target a tumor or to differentiate organs by colors, as presented in Figure 12.

Application

Here, a real case is considered where the previously introduced liver is deformed. The experiment is presented in Figure 13. Three points are fixed and a displacement is imposed onto the surface. The liver size is close to $100 \times 100 \times 200$ mm^3, and the displacement is approximately 40 mm. The employed Hookean material uses a Young's Modulus to 100 KPa and two different Poisson's ratio (μ=0.3 and 0.4). Note that the imposed displacement results in quite large deformations (13-18%).

Firstly, results obtained with our method are compared to certified numerical results. Then computational times are analyzed.

Accuracy

In this section, we compare accuracy at equilibrium. Resulting positions are compared to numerical results of classical approaches. These classical approaches need a volumetric mesh; the tetrahedral meshes presented in Figure 3 are employed.
The reference method is computed with the Ansys software [46], well known in engineering, by using the non-linear Green-Lagrange strain tensor (large displacement framework). Note that Ansys is not well adapted to interactive simulations, but its accuracy is still of interest.
A small displacement framework is also considered, computed by Ansys software using

the linear Cauchy strain tensor ("Ansys Cauchy Tetrahedra"). The co-rotational tetrahedra [19] ("Co-rotational Tetrahedra") is included in the comparison since a co-rotational approach is also employed for the hexahedral bounding elements. Moreover, it permits a speed comparison with another dynamic approach.

The comparison measurement is based on an Euclidean distance, node by node, between resulting positions of the evaluated methods and certified results (*i.e.,* Ansys using Green-Lagrange). To permit a good comparison, not only surface vertices positions but also all tetrahedra nodes (*i.e.,* inside nodes) positions are compared (referred to "Tetrahedral mesh"). Note that this imposes an interpolation of many more nodes than necessary with bounding elements, which normally need only a surface, in order to evaluate well the interior. A mean distance is computed, as well as the minimal and maximal distances, and the standard deviation. Results presented in Table 2 are given in millimeters. The experiment is done for two different tetrahedral resolutions.

It is difficult to compare computation time, since Ansys uses a static solver, whereas, bounding elements are solved dynamically. The equilibrium solution of the dynamical approaches is obtained after waiting stabilization.

Tetrahedral Resolution	Mechanics	Mean Error	Min	Max	Std Dev	FPS
Coarse^{r1} $\mu = .3$	Coarse Bounding El^{m1}	0.64	0.05	2.74	0.5	70
	Fine Bounding El^{m2}	0.61	0.07	1.95	0.38	15
	Co-rotational Tetra^{t1}	0.29	0.01	1.08	0.29	40
	Ansys Cauchy Tetra^{t1}	1.96	0.08	6.42	1.55	-
Fine^{r2} $\mu = .3$	Coarse Bounding El^{m1}	0.65	0.04	1.84	0.35	43
	Fine Bounding El^{m2}	0.98	0.04	1.7	0.36	13
	Co-rotational Tetra^{t2}	0.07	0	0.27	0.07	5
	Ansys Cauchy Tetra^{t2}	2.09	0.04	7.4	1.77	-
Coarse^{r1} $\mu = .4$	Coarse Bounding El^{m1}	0.8	0.05	3.24	0.73	70
	Fine Bounding El^{m2}	0.66	0.06	2.4	0.49	15
	Co-rotational Tetra^{t1}	0.24	0.03	0.77	0.19	40
	Ansys Cauchy Tetra^{t1}	1.94	0.1	6.36	1.54	-
Fine^{r2} $\mu = .4$	Coarse Bounding El^{m1}	0.64	0.04	1.89	0.3	43
	Fine Bounding El^{m2}	1.15	0.04	2.18	0.42	13
	Co-rotational Tetra^{t2}	0.01	0	0.02	0	5
	Ansys Cauchy Tetra^{t2}	2.08	0.03	7.12	1.73	-

Table 2. ^{r1} = 229 vertices, 452 triangles, ^{r2} = 1455 vertices, 2904 triangles , ^{m1} = 436 particles, 242 hexahedra, ^{m2} = 2003 particles, 1346 hexahedra, ^{t1} = 309 particles, 929 tetrahedra, ^{t2} = 2032 particles, 6725 tetrahedra

As expected, a linear strain calculation gives much more approximate results, whereas the co-rotational method approaches the Green-Lagrange tensor. Results for bounding elements are good, since the mean error remains below 1% (comparing to liver size). When

16   *Nesme M., Faure F., Payan Y.*

the resolution of the bounding elements increases, we were surprise to find the quality of the results decrease. This could be the result of the system becoming larger and the iterative solver, with a limited number of iterations, finishes before convergence resulting in an overly approximated solution. For an accuracy almost similar, the speedup is not negligible compared to traditional co-rotational tetrahedra, that are already very fast. In our opinion, this order of magnitude, even with few mechanical degrees of freedom, makes bounding elements suitable for interactive animation where a certain accuracy is required, like training surgical simulators.

Performance

Figure 14 gives results for several simulations that were run on a laptop Pentium M 2 GHz with 2 GB of RAM and a nVidia Quadro FX Go1400.

Described here are the results of external forces applied to the liver. We found equivalent results for various complex objects.

Two sets of simulations were tested. In the first, the mechanical resolution was varied. In the second, the geometrical resolution was varied. Resolution values are presented in Table 3. The number of implicit iterations was fixed at 10 for these examples.

Computation time is presented in Figure 14. Performance depend both on the number of interpolated points, *i.e.,* surface points, and the number of mechanical nodes. This is a great advantage, since it is possible to keep a fast framerate even when a high detailed mesh is animated, by reducing the number of elements. It should be noted that when the mechanical mesh is coarse, the rendering resolution has a signifiant impact on the framerate. However, when the mechanical mesh is more detailed the amount of time required for computation is greater than that needed to render a fine geometrical model, therefor its overall influence on the framerate becomes negligible.

mechanical resolutions		
name	#nodes	#elements
m0	27	8
m1	107	46
m2	436	242
m3	2003	1346
m4	11378	8913

geometrical resolutions		
name	#points	#triangles
g0	229	452
g1	1455	2904
g2	5541	11078

Table 3. Sets of tests.

In our implementation, a large amount of time is used to update surface vertices. It would be possible to accelerate this by performing the interpolations on the GPU using a vertex shader. Preliminary results show that a very simple shader that only computes final

interpolations for rendering, without reading back the results for collision management, accelerates the most geometrically detailed animation by a factor of two.

Previous results used OpenGL with a triangle-based rendering. The liver presented in figure 11 with volumetric rendering runs at 10 fps using 291 bounding elements and 482 particles.

Conclusion

We proposed a novel technique for directly animating a soft object using finite elements without passing by complex steps of volume meshing like tetrahedrization. A nice feature of our method is its ability to animate directly objects obtained from volumetric data or from a surface mesh. Since voxels are built automatically inside the object and around the surface, complex steps to build a volumetric mesh are no longer necessary. Our method is therefore very well adapted for object or patient specific applications. Since mechanical complexity is decoupled from geometrical detail, the method is well suited for interactive applications such as training simulators. A reduced number of degrees of freedom can be used for representing the mechanics while maintaining a highly detailed rendering. Additional data computed in a preprocessing stage (grid points, interpolation weights and stiffness matrices) allows a fast and robust application of the finite element dynamics. The resulting complexity is comparable to that of traditional FEM. Moreover, accounting for the distribution of matter in the proposed manner results in a more physically plausible behaviour along the boundary cells of the objects. Since the method is intended for interactive simulators, the goal is to compute as accurate as possible deformations during the assigned time. Tests on several experiences showed that the approximated deformations appear realistic even if computation time is very short. Finally, interactive animation of rendered volumetric data is achieved using the simulation grid, which is well adapted to volumetric rendering.

Currently, a linear interpolation of the object inside cells is performed, which can introduce discontinuities of normals in the case of large deformations. Interpolation that is more continuous could be an improvement, for example by using Bernstein polynomials.
All the examples use a single mechanical level, as future work, the precomputed non-uniform properties could be extended to meshes with non-uniform resolution, such as deformable octrees.

To summarize, the proposed method is faster than co-rotational tetrahedra for a similar accuracy, with the added advantage of avoiding meshing problems and with the possibility of including extra information, like voxel gray scale based stiffness values.

Acknowledgments

18   *Nesme M., Faure F., Payan Y.*

ertier for his help on Ansys simulations and to Midori Hyndman for her valuable advice in English.

Appendix

Principles of Virtual Works

Since the virtual work has to be equal on a surface point p and on its lattice points q:

$$f(q)^T V(q) = f(p)^T V(p)$$

where f is the virtual force and V the virtual velocity.

$$f(q)^T = f(p)^T V(p) V^{-1}(q) = f(p)^T H V(q) V^{-1}(q) = f(p)^T H$$

with H the interpolation matrix from q to p ($u(p) = Hu(q)$, *cf* eq. (1)). So

$$f(q) = H^T f(p)$$

Interpolation Matrices for Hexahedral Subdivision

The following presents the eight interpolation matrices L_c that give the values of the child cell c nodes from their parent cell nodes, in the order indicated in Figure 15. These values are very easy to compute. If $a_{i \to c_j}$ is the influence of the ith parent point on the jth point of its cth child, then

$$\mathbf{L}_c = \begin{bmatrix} a_{c_00} & ... & a_{c_07} \\ & ... & \\ a_{c_70} & ... & a_{c_77} \end{bmatrix}$$

$$\mathbf{L}_0 = \frac{1}{8}\begin{bmatrix} 8&0&0&0&0&0&0&0 \\ 4&4&0&0&0&0&0&0 \\ 4&0&4&0&0&0&0&0 \\ 2&2&2&2&0&0&0&0 \\ 4&0&0&0&4&0&0&0 \\ 2&2&0&0&2&2&0&0 \\ 2&0&2&0&2&0&2&0 \\ 1&1&1&1&1&1&1&1 \end{bmatrix} \quad \mathbf{L}_1 = \frac{1}{8}\begin{bmatrix} 4&4&0&0&0&0&0&0 \\ 0&8&0&0&0&0&0&0 \\ 2&2&2&2&0&0&0&0 \\ 0&4&0&4&0&0&0&0 \\ 2&2&0&0&2&2&0&0 \\ 0&4&0&0&0&4&0&0 \\ 1&1&1&1&1&1&1&1 \\ 0&2&0&2&0&2&0&2 \end{bmatrix}$$

$$\mathbf{L}_2 = \frac{1}{8}\begin{bmatrix} 4&0&4&0&0&0&0&0 \\ 2&2&2&2&0&0&0&0 \\ 0&0&8&0&0&0&0&0 \\ 0&0&4&4&0&0&0&0 \\ 2&0&2&0&2&0&2&0 \\ 1&1&1&1&1&1&1&1 \\ 0&0&4&0&0&0&4&0 \\ 0&0&2&2&0&0&2&2 \end{bmatrix} \quad \mathbf{L}_3 = \frac{1}{8}\begin{bmatrix} 2&2&2&2&0&0&0&0 \\ 0&4&0&4&0&0&0&0 \\ 0&0&4&4&0&0&0&0 \\ 0&0&0&8&0&0&0&0 \\ 1&1&1&1&1&1&1&1 \\ 0&2&0&2&0&2&0&2 \\ 0&0&2&2&0&0&2&2 \\ 0&0&0&4&0&0&0&4 \end{bmatrix}$$

$$\mathbf{L}_4 = \frac{1}{8}\begin{bmatrix} 4&0&0&0&4&0&0&0 \\ 2&2&0&0&2&2&0&0 \\ 2&0&2&0&2&0&2&0 \\ 1&1&1&1&1&1&1&1 \\ 0&0&0&0&8&0&0&0 \\ 0&0&0&0&4&4&0&0 \\ 0&0&0&0&4&0&4&0 \\ 0&0&0&0&2&2&2&2 \end{bmatrix} \quad \mathbf{L}_5 = \frac{1}{8}\begin{bmatrix} 2&2&0&0&2&2&0&0 \\ 0&4&0&0&0&4&0&0 \\ 1&1&1&1&1&1&1&1 \\ 0&2&0&2&0&2&0&2 \\ 0&0&0&0&4&4&0&0 \\ 0&0&0&0&0&8&0&0 \\ 0&0&0&0&2&2&2&2 \\ 0&0&0&0&0&4&0&4 \end{bmatrix}$$

$$\mathbf{L}_6 = \frac{1}{8}\begin{bmatrix} 2&0&2&0&2&0&2&0 \\ 1&1&1&1&1&1&1&1 \\ 0&0&4&0&0&0&4&0 \\ 0&0&2&2&0&0&2&2 \\ 0&0&0&0&4&0&4&0 \\ 0&0&0&0&2&2&2&2 \\ 0&0&0&0&0&0&8&0 \\ 0&0&0&0&0&0&4&4 \end{bmatrix} \quad \mathbf{L}_7 = \frac{1}{8}\begin{bmatrix} 1&1&1&1&1&1&1&1 \\ 0&2&0&2&0&2&0&2 \\ 0&0&2&2&0&0&2&2 \\ 0&0&0&4&0&0&0&4 \\ 0&0&0&0&2&2&2&2 \\ 0&0&0&0&0&4&0&4 \\ 0&0&0&0&0&0&4&4 \\ 0&0&0&0&0&0&0&8 \end{bmatrix}$$

Triangulation of a Planar Convex Polygon

This triangulation is shown in Figure 16. Vertices are separated in four groups depending on the sign of cosine and sine of their angle with a line passing through a random vertex P_0. The resulting triangle-strip starts with P_0 and traverse the four groups in the right order using cosine and sine values in each group.

References

1. William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, 1987.
2. T.J.R. Hugues. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Prentice-Hall, 1987.
3. B. Couteau, Y. Payan, and S. Lavallée. The mesh-matching algorithm: an automatic 3d mesh generator for finite element structures. In *Journal of Biomechanics*, volume 33/8, pages 1005–1009, 2000.
4. A. Castellano-Smith, T. Hartkens, J. Schnabel, D. Hose, H. Liu, W. Hall, C. Truwit, D. Hawkes, and D. Hill. Constructing patient specific models for correcting intraoperative brain deformation. In *Lecture Notes in Computer Science*, volume 2208, pages 1091–1098, 2001.
5. O. Clatz, M. Sermesant, P.-Y. Bondiau, H. Delingette, S. K. Warfield, G. Malandain, and N. Ayache. Realistic simulation of the 3d growth of brain tumors in mr images including diffusion and mass effect. In *IEEE Transactions on Medical Imaging*, volume 24(10), pages 1334–1346, 2005.
6. J. H. Keyak, J. M. Meagher, H. B. Skinner, and C. D. Mote Jr. Automated three-dimensional finite element modelling of bone: a new method. In *Journal of Biomedial Engineering*, volume 12, pages 389–397, September 1990.
7. Y. Su, K.H. Lee, and A. Senthil Kumar. Automatic hexahedral mesh generation for multi-domain composite models using a hybrid projective grid-based method. In *Computer-Aided Design*, volume 36(3), pages 203–215, 2004.
8. Marco Viceconti, Mario Davinelli, Fulvia Taddei, and Angelo Cappello. Automatic generation of accurate subject-specific bone finite element models to be used in clinical studies. In *Journal of Biomechanics*, volume 37, pages 1597–1605, 2004.
9. Claudio Lobos, Marek Bucki, Nancy Hitschfeld, , and Yohan Payan. Mixed-element mesh for an intra-operative modeling of the brain tumor extraction. In *16th International Meshing Roundtable*, 2007.
10. K.J. Bathe. *Finite Element Procedures in Engineering Analysis*. Prentice-Hall, 1982.
11. Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *SIGGRAPH '87*, 1987.
12. Steven Cover, Norberto Ezquerra, James O'Brien, Richard Rowe, Thomas Gadacz, and Ellen Palm. Interactively deformable models for surgery simulation. *IEEE Comput. Graph. Appl.*, 13(6):68–75, 1993.
13. David E. Breen, Donald H. House, and Michael J. Wozny. Predicting the drape of woven cloth using interacting particles. In *Proc SIGGRAPH'94*, pages 365–372, New York, NY, USA, 1994. ACM Press.
14. Michael Hauth. Visual simulation of deformable models. Phd thesis, Wilhelm-Schickard-Institut für Informatik, University of Tübingen, Germany, July 2004.
15. M. Müller, J. Dorsey, L. McMillan, R. Jagnow, and B. Cutler. Stable real-time deformations. In *Proc SCA*, 2002.

16. O. Etzmuß and M. Keckeisen. A Linearised Finite Element Model for Cloth Animation. Technical Report WSI-2003-2, Universität Tübingen, 2003.
17. M. Müller and M. Gross. Interactive virtual materials. In *Proc Graphics Interface*, 2004.
18. G. Irving, J. Teran, and R. Fedkiw. Invertible finite elements for robust simulation of large deformation. In *Proc SCA*, 2004.
19. M. Nesme, Y. Payan, and F. Faure. Efficient, physically plausible finite elements. In *Eurographics (short papers)*, pages 77–80, august 2005.
20. G. Debunne, M. Desbrun, M-P. Cani, and A. H. Barr. Dynamic real-time deformations using space and time adaptive sampling. In *SIGGRAPH '01*, 2001.
21. Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. A multiresolution framework for dynamic deformations. In *SCA '02*, 2002.
22. Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. Interactive skeleton-driven dynamic deformations. In *Proc SIGGRAPH'02*, 2002.
23. J. Mosegaard and T.S. Sørensen. Real-time deformation of detailed geometry based on mappings to a less detailed physical simulation on the gpu. In *11th Eurographics Workshop on Virtual Enviroments*, pages 105–111, 2005.
24. M. Müller, M. Teschner, and M. Gross. Physically based simulation of objects represented by surface meshes. In *Proc SCA*, 2004.
25. Doug L. James, Jernej Barbič, and Christopher D. Twigg. Squashing cubes: Automating deformable model construction for graphics. In *Proc SIGGRAPH'04 Conference on Sketches & Applications*, 2004.
26. J. Dequidt, D. Marchal, and L. Grisoni. Time critical animation of deformable solids. *Journal of Computer Animation and Virtual Worlds*, 16:177–187, 2005.
27. Matthieu Nesme, François Faure, and Yohan Payan. Hierarchical multi-resolution finite element model for soft body simulation. *Lecture Notes in Computer Science (Proc. of Symposium on Biomedical Simulation)*, 4072:40–47, 2006.
28. Jernej Barbič and Doug L. James. Real-time subspace integration for st.venant-kirchhoff deformable models. *ACM Transactions on Graphics (SIGGRAPH 2005)*, 24(3), August 2005.
29. M. Müller, B. Heidelberger, M. Teschner, and M. Gross. Meshless deformations based on shape matching. In *Proc SIGGRAPH'05*, pages 471–478, july 2005.
30. M. Desbrun and M.-P. Cani. Animating soft substances with implicit surfaces. In *Proc SIGGRAPH'95*, pages 287–290, 1995.
31. M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *Proc SCA*, pages 141–151, 2004.
32. Mark Pauly, Richard Keiser, Bart Adams, Philip Dutré;, Markus Gross, and Leonidas J. Guibas. Meshless animation of fracturing solids. *ACM Trans. Graph.*, 24(3):957–964, 2005.
33. Press, Teukolski, Vetterling, and Flannery. *Numerical Recipes in C*. Cambridge University Press, 1992.
34. David Baraff and Andrew Witkin. Large steps in cloth simulation. In *SIGGRAPH '98*, 1998.
35. M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, Laks Raghupathi, A. Fuhrmann, Marie-Paule Cani, François Faure, N. Magnetat-Thalmann, W. Strasser, and P. Volino. Collision detection for deformable objects. *Computer Graphics Forum*, 24(1):61–81, March 2005.
36. R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *SCA*, 2003.
37. Jianlong Zhou and Klaus D. Tonnies. State of the art for volume rendering. Technical report, Institute for Simulation and Graphics, University of Magdeburg, Germany, 2003.
38. Alexandre Meyer and Fabrice Neyret. Interactive volumetric textures. In *Rendering Techniques (Eurographics Workshop on Rendering - EGSR)*, pages 157–168, Jul 1998.
39. Hendrik P. A. Lensch, Katja Daubert, and Hans-Peter Seidel. Interactive semi-transparent volumetric textures. In *Proc Vision, Modeling and Visualization*, pages 505–512, 2002.

40. Ashley Horton, Adam Wittek, and Karol Miller. Towards meshless methods for surgical simu-
lation. In *MICCAI Workshop*, 2006.

41. Matthieu Nesme, Yohan Payan, and François Faure. Animating shapes at arbitrary resolution
with non-uniform stiffness. In *Eurographics Workshop in Virtual Reality Interaction and Phys-
ical Simulation (VRIPHYS)*, 2006.

42. Hang Si. Tetgen, a quality tetrahedral mesh generator and three-dimensional delaunay triangu-
lator.

43. O. Etzmuß, M. Keckeisen, and W. Straßer. A Fast Finite Element Solution for Cloth Modelling.
Proc Pacific Graphics, 2003.

44. William M. Hsu, John F. Hughes, and Henry Kaufman. Direct manipulation of free-form defor-
mations. In *Proc of SIGGRAPH '92*, pages 177–184, New York, NY, USA, 1992. ACM Press.

45. Matthieu Nesme, Maud Marchal, Emmanuel Promayon, Matthieu Chabanas, Yohan Payan, and
François Faure. Physically realistic interactive simulation for biological soft tissues. *Recent Re-
search Developments in Biomechanics*, 2, 2005.

46. Ansys, engineering simulation software.

47. Philip Schneider and David H. Eberly. *Geometric Tools for Computer Graphic*. Morgan Kauf-
mann Publishers, Inc., 2003.
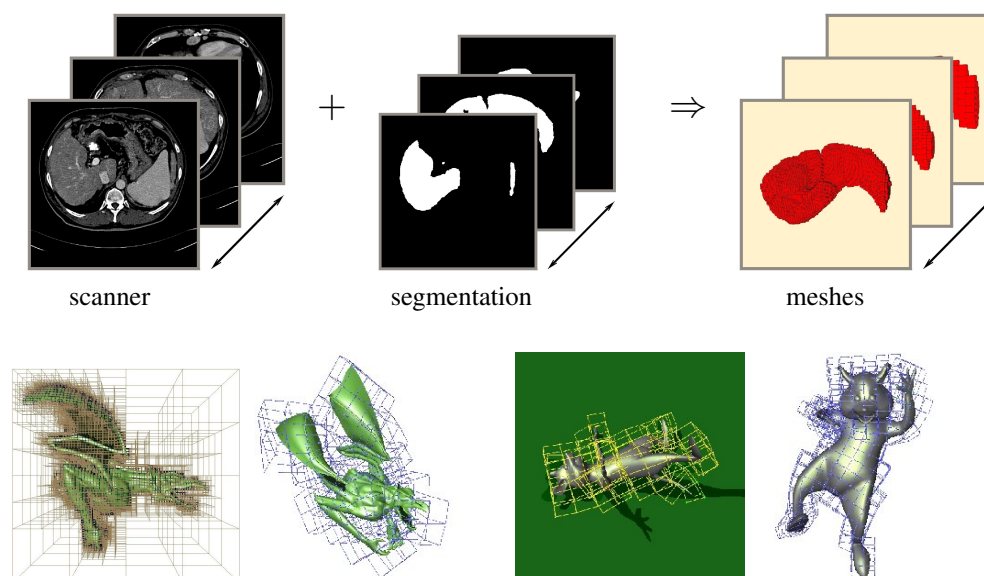
Fig. 1. Building hexahedral meshes at several resolutions. *Top:* from scanner data and segmented pictures (here
for a liver). *Bottom:* from complex surface meshes.

Accurate Interactive Animation of Deformable Models at Arbitrary Resolution   23
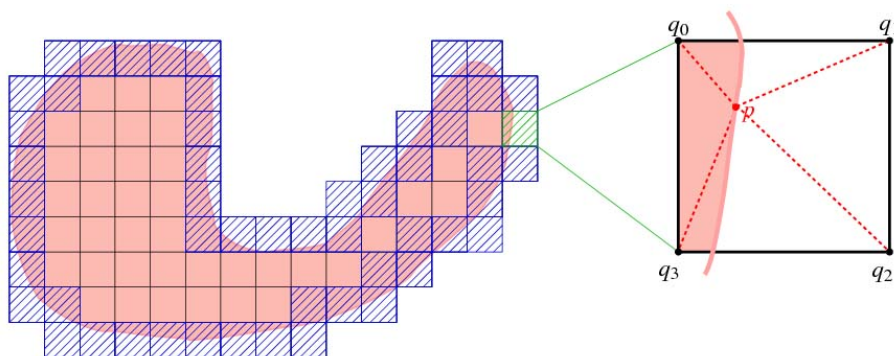
Fig. 2. Animated object is interpolated inside deformable bounding elements.
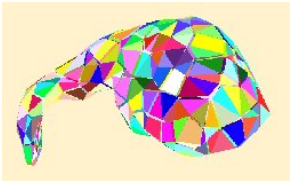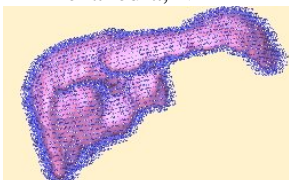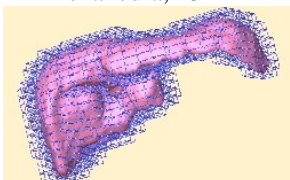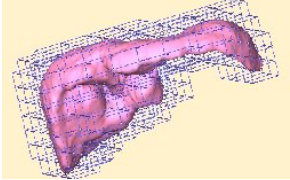
original marching cube 5541 surface vertices, 11078 triangles	≈ 70% decimation 1455 surface vertices, 2904 triangles	≈ 95% decimation 229 surface vertices, 452 triangles
7655 nodes, 25204 tetrahedra, 1 Hz	2032 nodes, 6725 tetrahedra, 5 Hz	309 nodes, 929 tetrahedra, 40 Hz
11378 nodes, 8913 hexahedra, <1 Hz	2003 nodes, 1346 hexahedra, 13 Hz	436 nodes, 242 hexahedra, 70 Hz

Fig. 3. Several mechanical mesh resolutions for classical tetrahedral meshes (top) and bounding hexahedral meshes (bottom) with animation rate. Non-realistic rendering of tetrahedra (left) permits to well distinguish them.

24   *Nesme M., Faure F., Payan Y.*
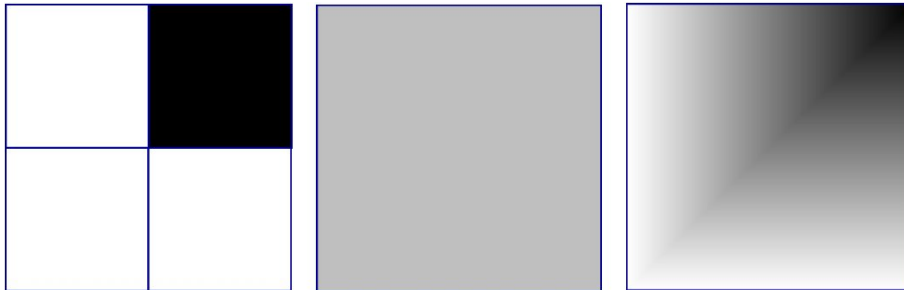
Fig. 4. 2D stiffness in level of gray. From left to right: *a*) four original elements, three empty, one full. *b*) the corresponding element with a classical uniform weighted behavior. *c*) the corresponding non-uniform element, stiffness varies along the axes
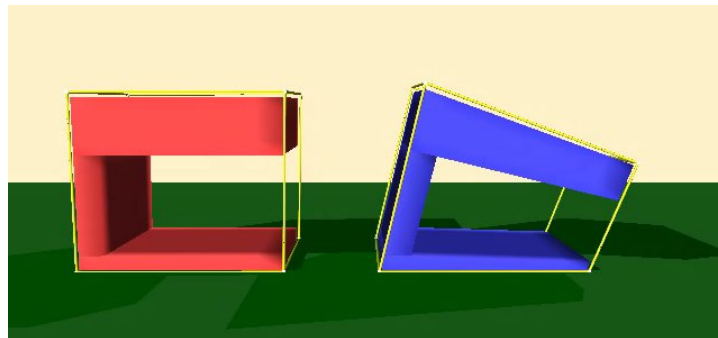
Fig. 5. Two similar objects are simulated by one single cell and subject to gravity. The left one is simulated using a basic uniform law and the right one, using the precomputed non-uniform law.
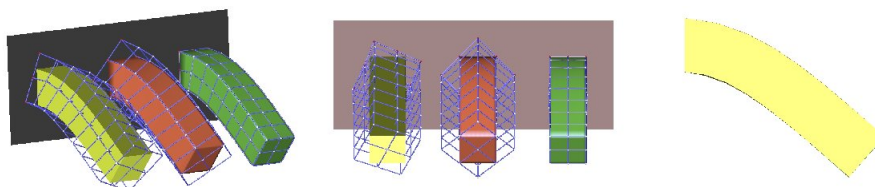
Fig. 6. Bounding elements of a fixed beam placed in best, *i.e.,* classical, case (green, right position) vs voluntarily placed in bad cases (red, middle and yellow, left position). (The right picture superimposes results for all cases)
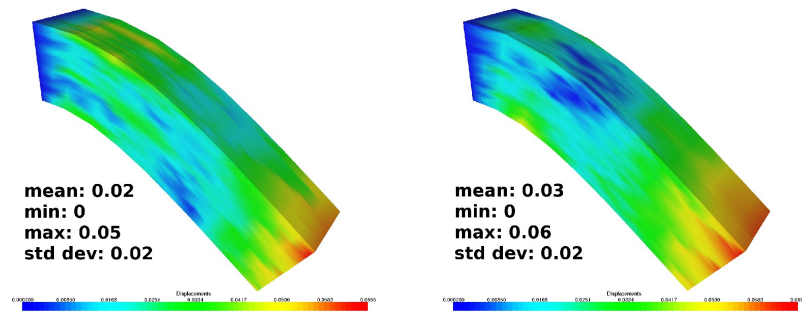
Fig. 7. Euclidean distance error (in mm) versus well-fitted hexahedra. Left: yellow beam, Right: red beam of Figure 6
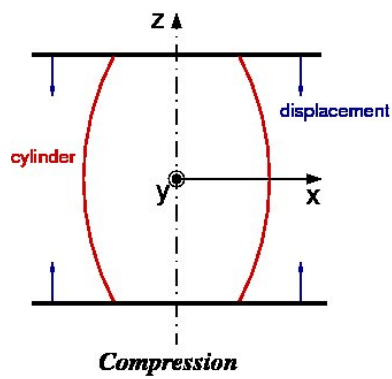
Fig. 8. Compression of a cylinder composed of quasi-incompressible material.

26   *Nesme M., Faure F., Payan Y.*
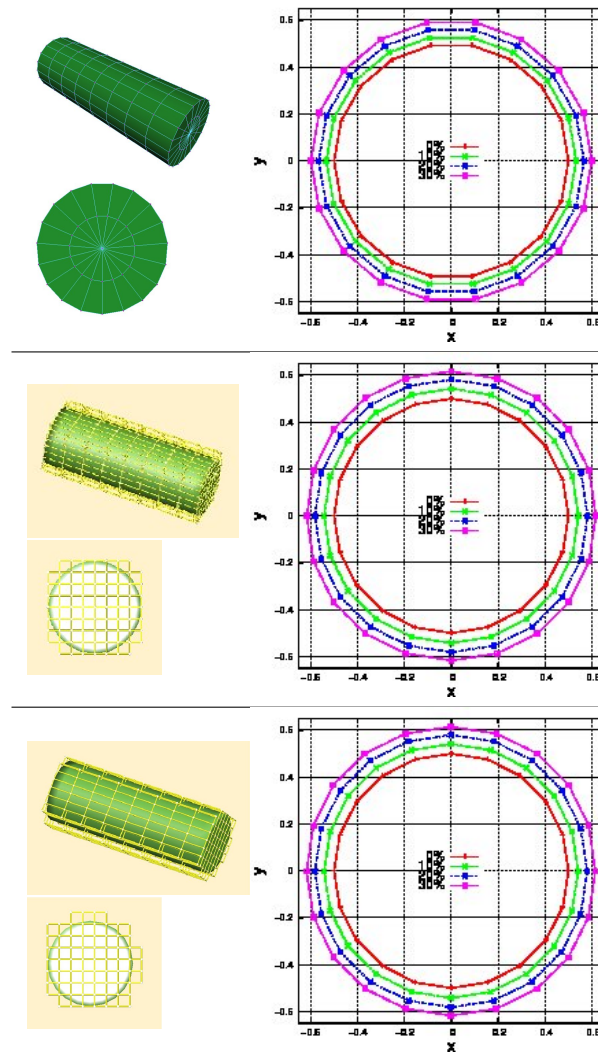
Fig. 9. Meshed cylinders. top: classical mesh using prisms and hexahedra. middle: bounding elements well fitted. down: bounding elements bad fitted. right column: Circumference of the median plane of the cylinder at rest form and after several compressions.
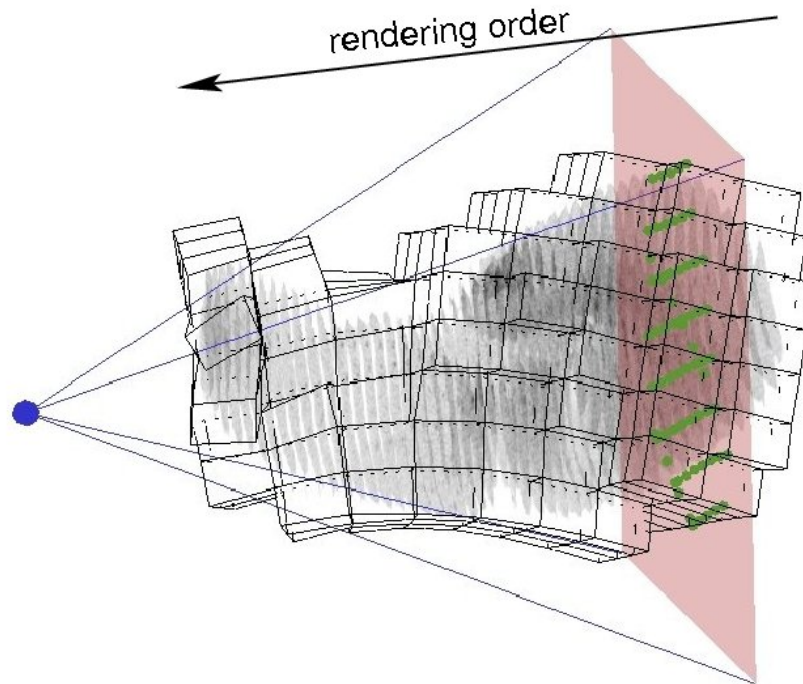
Fig. 10. A slice (in red), orthogonal to the view point (in blue) cuts a deformed hexahedral mesh (in black). All intersections between the slice and all edges are represented by green dots. Triangles are rendered in semi-transparency from back to front. Texture coordinates at intersection points can be interpolated between hexahedra corners.

28   *Nesme M., Faure F., Payan Y.*
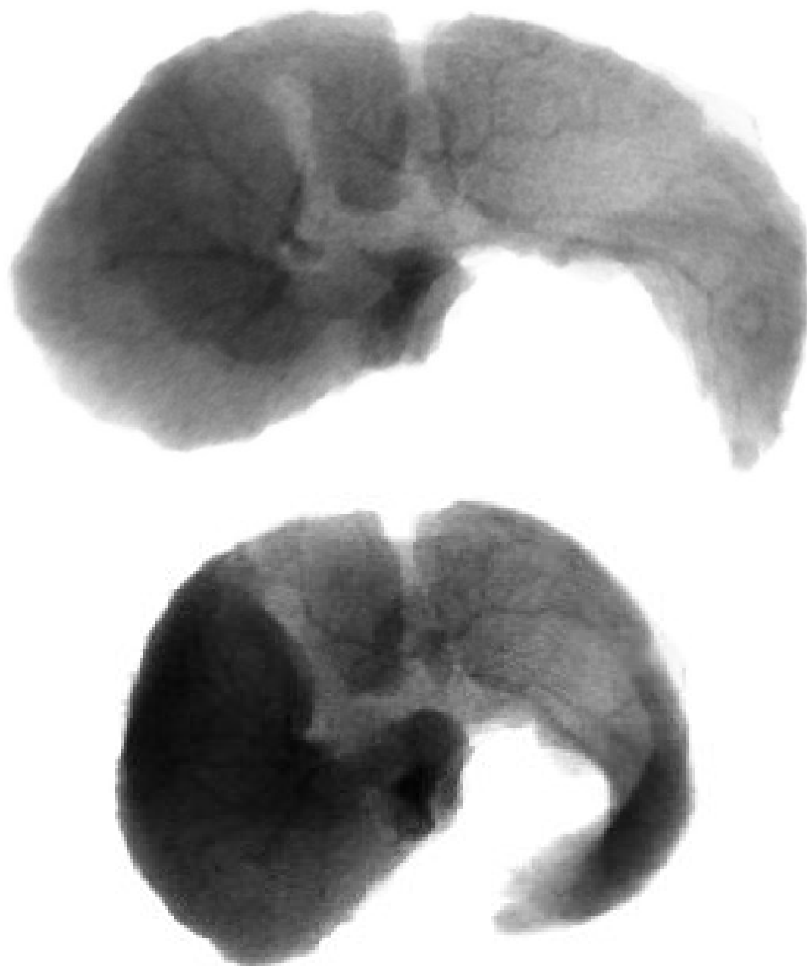
Fig. 11. The volumetric rendering of a deformed liver.

Fig. 12. Top: a tumor is targeted in red. Bottom: several organs located by different colors (heart in red, liver in pink, piece of left lung in blue, bladder in green)
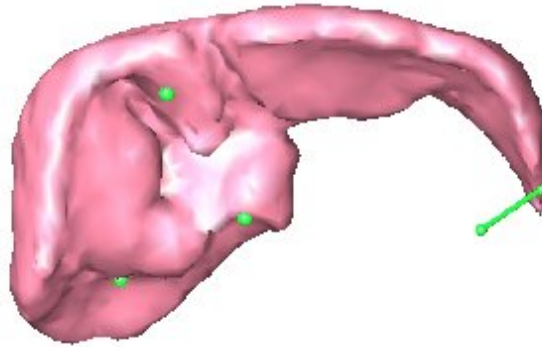
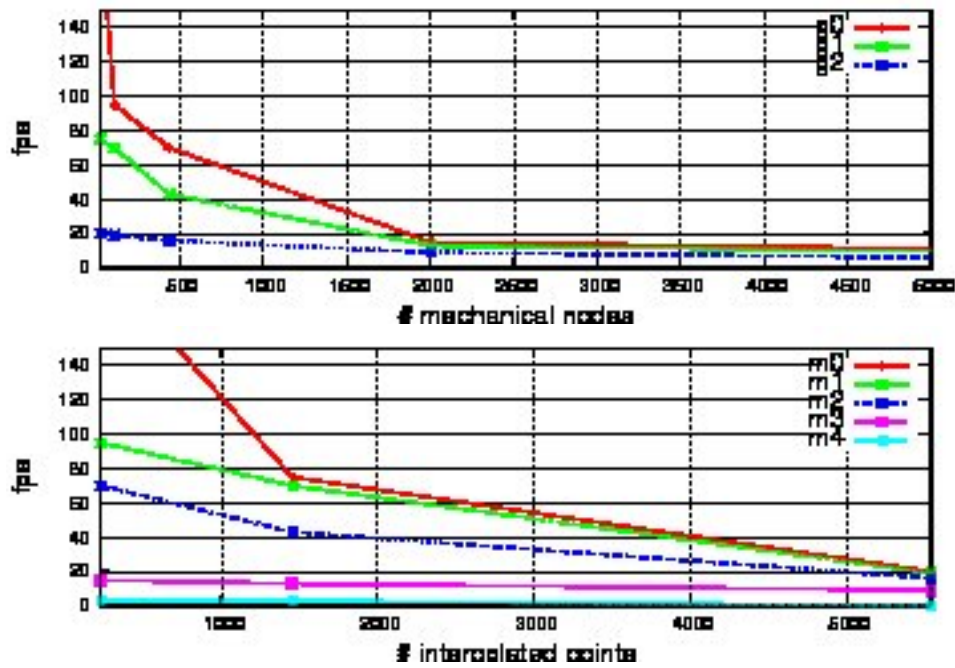Fig. 13. The experiment: three nodes are fixed and a displacement is imposed on a fourth one.

Fig. 14. Speed results in rendered frames per second using sets of tests presented on table 3. top: depending on mechanical resolution. down: depending on geometric resolution.
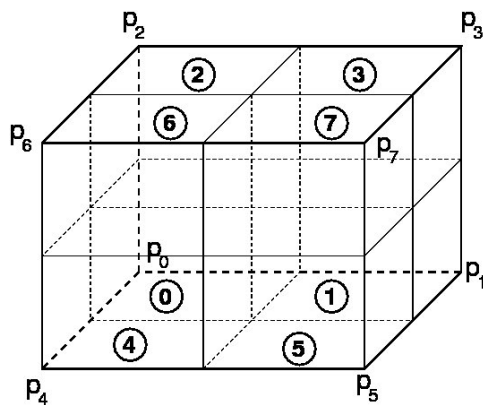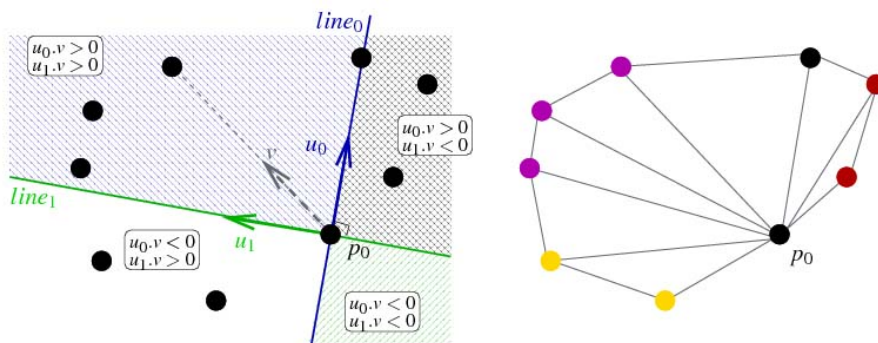
Fig. 15. Index for hexahedral subdivision.

Fig. 16. Triangulation of a planar convex polygon.

Photo and Bibliography

Matthieu Nesme is currently a post-doctoral fellow at McGill University, Montréal. He completed a Ph.D. in Computer Science at Grenoble Universités in 2008 and he received the master degree in "image, vision, robotics" from Institut National Polytechnique of Grenoble in 2004. His research interests concern physical-based animation including soft body modeling and collision processing.

François Faure completed a Ph.D. thesis in Computer Science at University Joseph Fourier, Grenoble, France in 1997. He previously studied Mechanical Engineering at the Ecole Normale Supérieure de Cachan. His research interests are mostly about the interactive physically based animation of complex scenes. François is an Assistant Professor at LJK computer graphics lab, CNRS, INRIA, University of Grenoble I.

Yohan Payan received the Ph.D. degree in computer science and signal processing, from the Institut National Polytechnique of Grenoble, Grenoble, France, in 1996. Currently, he is researcher at the French National Research Institute (CNRS), La Tronche, France, in the TIMC-IMAG laboratory, with interests that include biomechanics, computer-assisted surgery, signal processing, and neurosciences.