Soft Tissue Modeling for Surgery Simulation

Hervé Delingette, Nicholas Ayache

Soft Tissue Modeling for Surgery Simulation

Hervé Delingette and Nicholas Ayache

Contents

Foreword

In this chapter, we address a specific issue belonging to the field of biomechanics : modeling living tissue deformation with real-time constraints. This issue was raised by the emergence, in the middle of the 1990's, of a very specific application : the simulation of surgical procedures. This new concept of surgery simulation was in large part advocated by the American Department of Defense [93], for which this concept was a key part of their vision of the future of emergency medicine.

Since then, the concept of having surgeons being trained on simulators (just like pilots on flight simulators) has been refined. First, the development of commercial simulators has proved that there was a demand for products that help to optimize the learning curve of surgeons[1]. Second, the emergence of medical robotics and more precisely of minimally invasive surgery robots, has reinforced the need for simulating surgical procedures, since these robots require a very specific hand-eye coordination. Third, there is a large consensus among the medical community that current simulators are not realistic enough to provide advanced gesture training. In particular, the modeling of living tissue, and their ability to deform under the contact of an instrument is one of the important aspect of simulators that should be improved.

In this chapter, we present different algorithms for modeling soft tissue deformation in the context of surgery simulation. These algorithms make radical simplifications about tissue material property, tissue visco-elasticity and tissue anatomy. The first section of this chapter describes the principles and the components of a surgical simulator. In particular, we insist on the different constraints of soft tissue models in this application, the most important being the real-time computation constraint. In section 2, we present the process of building a patient-specific hepatic surgery simulator from a set of medical images. The different stages of computation leading to the creation of a volumetric tetrahedral mesh from a medical image are especially emphasized. In section 3, we detail the five main hypotheses that are made in the proposed soft tissue models. Furthermore, we recall the main equations of isotropic and transversally anisotropic linear elasticity in continuum mechanics. The discretization of these equations are presented in section 4 based on finite element modeling. Because we rely on the simple linear tetrahedron element, we provide closed form expressions of local and global stiffness matrices. After describing the types of boundary conditions existing in surgery simulation, we derive the static and dynamic equilibrium equations in their matrix form. In section 5, a first model of soft tissue is proposed. It is based on the off-line inversion of the stiffness matrix, and can be computed very efficiently as long as no topology change is required. In such case, in section 6, a second soft tissue model allows to perform cutting and tearing but with less efficiency as the previous model. A combination of the two previous models, called "hybrid model" is also presented in this section. Finally, in section 7, we introduce an extension of the second soft tissue model that implements large displacement elasticity.

[1] This curve represents the number of incidents occuring during the performance of surgery as a function of time. This curve is generally monotonically decreasing under the effect of training and usually reaches an asymptotic value after a certain number N of real interventions. The objective of the simulators is to reduce this number N as much as possible.

1 General Issues in Surgery Simulation

1.1 Surgical Simulators

1.1.1 Medical Impact of Surgical Simulators

Surgery simulation aims at reproducing the visual and haptic senses experienced by a surgeon during a surgical procedure, through the use of computer and robotics systems. The medical interest of this technology is linked with the development of minimally invasive techniques especially video-surgery (endoscopy, laparoscopy,...). More precisely, laparoscopy consists in performing surgery by introducing different surgical instruments in the patient abdomen through one centimeter-wide incisions. The surgeon can see the abdominal anatomy with great clarity by watching a high resolution monitor connected to an endoscope introduced inside the patient abdomen. This technique bears several advantages over traditional open surgery. On one hand, it decreases the trauma entailed by the surgical procedure on the patient body. This allows to decrease the patient stay in hospitals and therefore decreases the cost of health care. On the other hand, it reduces the morbidity as demonstrated by the Hunter and Sackier study [51].

However, if these minimally invasive techniques are clearly beneficial to the patients, they also bring new constraints on the surgical practice. First, they significantly degrade the surgeon access to the patient body. In laparoscopy for instance, the surgical procedure is made more complex by the limited number of degrees of freedom of each surgical instrument. Indeed, they must go through fixed points where the incisions in the patient's abdomen were done. Furthermore, because the surgeon cannot see his hand on the monitor, this technique requires a specific hand-eye coordination. Therefore, an important training phase is required before a surgeon acquires the skills necessary to adequately perform minimally invasive surgery (corresponding to a plateau in the learning curve).

Currently, surgeons are trained to perform minimally invasive surgery by using mechanical simulators or living animals. The former method is based on "endotrainers" representing an abdominal cavity inside which are placed plastic objects representing human organs. These systems are sufficient for acquiring basic surgical skills but are not realistic enough to represent fully the complexity of the human anatomy and physiology (respiratory motion, bleeding,...). The latter training method consists in practicing simple or complex surgical procedures on living animals (often pigs for abdominal surgery). This method has two limitations. First, the similarity between the human and animal anatomy is limited and therefore certain procedures cannot be precisely simulated with this technique. Also, the evolution of the ethical code in most countries may forbid the use of animals for this specific training, as it is already the case in several European and North American countries.

Because of the limitations of current training methods, there is a large interest in developing video-surgery simulation software for providing efficient and quantitative gesture training systems [3]. Indeed, such systems should bring a greater flexibility by providing scenarios including different types of pathologies. Furthermore, thanks to the development of medical image reconstruction algorithms, surgery simulation allows surgeons to verify and optimize the surgical

strategy of a procedure on a given patient.

1.1.2 Classification of Surgical Simulators

Satava *et al.* [94] proposed to classify surgical simulators into three categories (see figure (1)). The first generation simulators are solely based on anatomical information, in particular on the geometry of the anatomical structures included in the simulator. In these simulators, the user can virtually navigate inside the human body but has a limited interaction with the modeled organs. Currently, several first generation surgical simulators are available including commercial products linked to medical imaging systems (CT or MRI scanners) that are focusing on virtual endoscopy (colonoscopy, tracheoscopy,...). In general, they are used as a complementary examination tools establishing a diagnosis (for instance when using virtual endoscopy) or as a surgical planning tool before performing surgery.
In addition to geometrical information, second generation simulators describe the physical properties of the human body. For instance, the modeling of soft tissue biomechanical properties enables the simulation of basic surgical gestures such as cutting or suturing. Currently, several prototypes of second generation simulators are being developed including the simulation of cholecystectomy [26, 59], of arthroscopy of the knee [43] and of gynecological surgery [105]. Section 2 will shortly describe the hepatic surgery simulator being developed at INRIA.

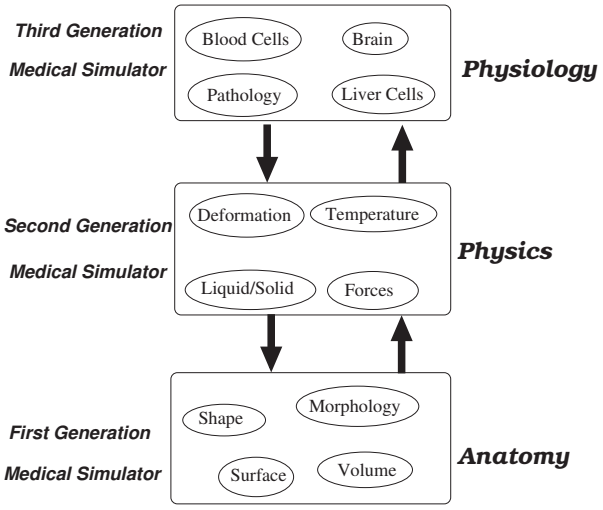


Figure 1: The different generations of medical simulators.

Third generation of surgical simulators provides an anatomical, physical and physiological description of the human body. There are very few simulators including these three levels of modeling, essentially because of the difficulty to realistically describe the coupling between physiology and physics. A good example of an attempt in this direction is given by the work of Kaye *et al.* [57] who modeled the mechanical cardiopulmonary interactions. Another important example is the study of the contraction of the right and left vectricles of the heart under the propagation of the action potential which is being carried out

by the group of Pr. McCulloch (this work is published in this book) but also by the INRIA ICEMA group [96, 97]. Finally, it should be noted that a comprehensive effort for creating computational physiological models has been recently launched in the international Physiome Project [5].

1.2 Simulator Architecture

In this section, we present the basic components of simulators for surgical gesture training and especially in the context of minimally invasive therapy. For the acquisition of basic skills, it is necessary to simulate the behavior of "living" tissues and therefore to develop a second generation surgical simulator. However, it raises important technical and scientific issues. The different components of these simulators are summarized in Figure (2).

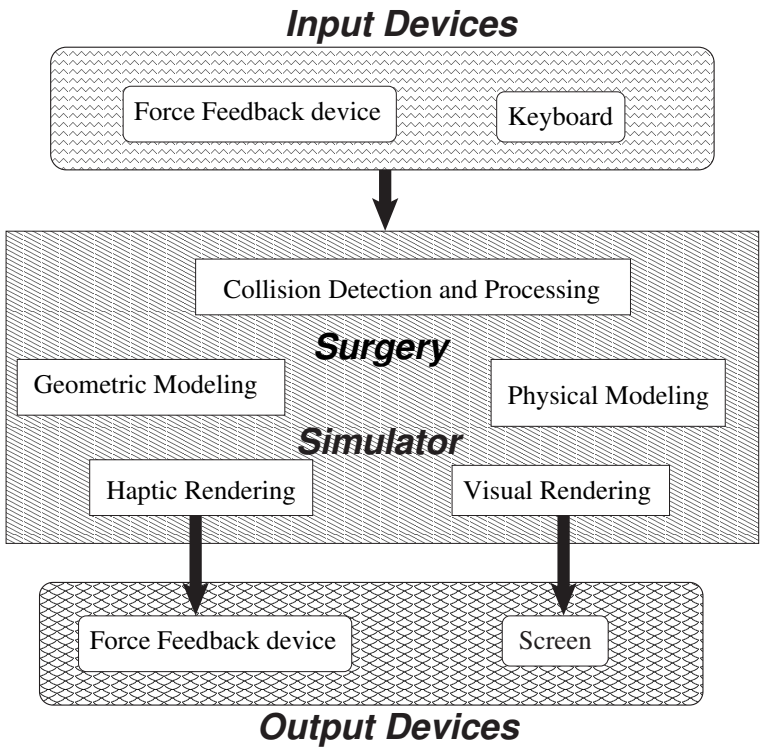


Figure 2: The different components of a second generation surgery simulator.

The input devices in such simulators usually consist of one or several mechanical systems that drive the motion of virtual surgical tools or of virtual endoscopes. In fact, as input devices they do not need to be motorized and they are usually equipped with simple optical encoders or position trackers. A keyboard and electronic mouse are also useful to modify the scenario of the simulation.
The core of a simulator consists of several modules. For instance, a first module provides the enabling tools for the creation of geometric models from medical images (see section 1.2.1). Another module, detailed in section 1.3, computes the deformation of soft tissues under the action of virtual instruments. These

interactions between virtual instruments and virtual organs, performed in a separate module, mainly consists in detecting collisions followed with modeling contact forces and displacements (see section 1.2.2).

Finally, a surgical simulator must provide an advanced user interface that includes visual and force feedback (respectively presented in sections 1.2.3 and 1.2.4). Last but not least, it is necessary to rely on advanced software engineering methodology to make these different modules communicate within the same framework : some of these implementation issues are introduced in section 1.2.5.

1.2.1 Geometric Modeling

In general, the extraction of tridimensional geometric models of anatomical structures are based on medical imagery: CT scanner images, MRI images, cryogenic images, 3D ultrasound images,... Because medical image resolution and contrast have greatly improved over the past few years, the tridimensional reconstruction of certain structures have become possible by using computerized tools. For instance, the availability in 1995 of the "Visible Human" dataset provided by the National Library of Medicine has allowed the creation of a complete geometric human model [1]. However, the automatic delineation of structures from medical images is still considered an unsolved problem. Therefore a lot of human interaction is usually required for reconstructing the human anatomy. In [35, 2], Duncan and Ayache provide a survey on the past and current research effort in medical image analysis.

1.2.2 Interaction with a Virtual Instrument

A key component of a surgery simulation software is the user interface. The hardware interface that drives the virtual instrument essentially consists in one or several force-feedback devices having the same degrees of freedom and appearance as the actual surgical instruments used in minimally invasive therapy (see figure (3)). In general, these systems are force-controlled, sending the instrument's position to the simulation software and receiving reaction force vectors.

Once the position of the virtual instrument is known, it is necessary to detect possible collisions with other instruments or surrounding anatomical structures. In this case, it is particularly difficult to obtain a computationally efficient collision detection algorithm because the geometry of objects may change at each iteration. Therefore, algorithms based on pre-computed data structures (such as the approach proposed in [45]) are not appropriate. In [64], Lombardo *et al.* proposed an original collision detection method based on the OpenGL graphics library which is especially well-suited for elongated instruments shaped like those used in laparoscopic surgery. Although this technique cannot be used for the detection of self-collisions, several algorithms have been proposed recently [106, 58] to tackle this complex task.

When a collision is detected, a set of geometrical or physical constraints are applied on soft tissue models. However, modeling the physics of contacts can lead to complex algorithms and therefore purely geometric approaches are often preferred.

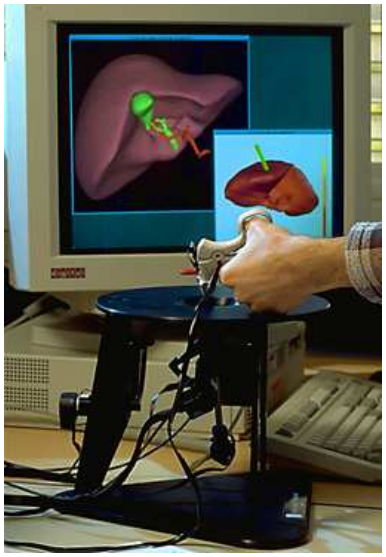


Figure 3: A force feedback system suited for surgery simulation

1.2.3 Visual Feedback

A surgery simulator must provide a realistic visual representation of the surgical procedure. Visual feedback is especially important in video-surgery because it helps the surgeon to acquire a tridimensional perception of his environment. In particular, the effects of shading, shadows and textures are important clues that must be reproduced in a simulator.
The quality of visual feedback is directly related to the availability and performance of graphics accelerators. In the past few years, the market of graphics cards has evolved in three directions : improved price-performance ratio, increased geometric transformation and rasterization performance and the emergence of programmable pixel rendering. Combined with the development of more efficient computer graphics algorithms, we can foresee that realistic visual feedback for surgery simulation could be achieved in the next few years if this additional graphics rendering is focused on the three-dimensional clues used by surgeons to understand the surgical field.

1.2.4 Haptic feedback

Haptic display serves at least two purposes in a surgical simulator : kinesthetic and cognitive. First, it provides the sensation of movement to the user and therefore it significantly enhances it surgical performance. Second, it helps to distinguish between tissues by testing their mechanical properties.
However, the addition of a haptic display in a simulation system increases by a large factor its complexity and the required computational power [71] : it leads to an increase by a factor 10 of the required bandwidth, synchronisation between visual and haptic displays, force computation, ... Only a few papers have assessed the importance of haptic feedback in surgery [70]. In general, it

is accepted that the combination of visual and haptic displays is optimal for surgery training or pre-planning.
In video-surgery, the surgical instruments slide inside a trocard and are constrained to enter the abdomen through a fixed point. This entails substantial friction, specifically for laparoscopy where airtightness must be enforced. The friction of the instruments inside trocards perturbes the forces sensed by the end-user. Despite those perturbations, it appears that it is still necessary to provide force-feedback for realistic user immersion.

1.2.5 Implementation of a simulator

Most of the difficulties encountered when implementing a surgical simulator originate from the trade-off that must be found between real-time interaction and the necessary surgical realism of a simulator.
The first constraint indicates that a minimum bandwidth between the computer and the interface devices must be available in order to provide a satisfactory visual and haptic feedback. If this bandwidth is too small, the user cannot properly interact with the simulator and it becomes useless for surgery gesture training. However, the "real-time" constraint can be interpreted in different ways. Most of the time, it implies that the mean update rate is high enough to allow a suitable interaction. However, it is possible that during the simulation, some events (such as the collision with a new structure) may increase the computational load of the simulation engine. This may result in a lack of synchronicity between the user gesture and the feedback the user gets from the simulator. When the computation time is too irregular, the user may even not be able to use the simulator properly. In order to guarantee a good user interaction, it is necessary to use a dedicated "real-time" software that supervises all tasks executed on the simulator.
The second constraint is related to the targeted application of a simulator : training surgeons to new gestures or procedures. To reach this goal, the user must "believe" that the simulator environment corresponds to a real procedure. The level of realism of a simulator is therefore very dependent on the type of surgical procedures and is also connected with physio-psychological parameters. In any case, increasing the realism of a simulator requires an increase of computational time which is contradictory with the constraint of real-time interaction. The main difficulty in implementing a simulator is to optimize its credibility, given an amount of graphics and computational resources. Therefore, an analysis of the training scenario should be performed to find the most important elements that contribute to the realism of the simulation.

1.3 Constraints of Soft Tissue Models

In the scope of a surgical simulator, it is not possible to model the biomechanical complexity of living soft tissue. Instead, authors have resorted to simplified models to decrease the implementation complexity and to optimize computational efficiency. A survey on soft tissue modeling can be found in [30].
Before presenting the main features of our approach (available in section 3.1), we list three constraints that should be taken into account when specifying a soft tissue model for surgery simulation.

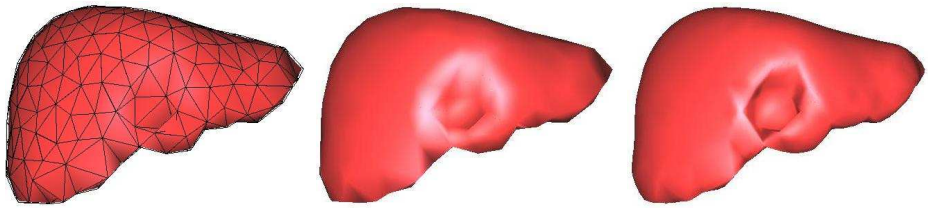
1.3.1 Visualization Constraints

To obtain high quality visual rendering, two techniques are traditionnally used : surface and volume rendering. A comparison between these two rendering techniques for surgery simulation is described in [100]. Surface rendering is by far the most commonly used technique, and uses basic polygonal elements (triangles, quads, ...) to achieve the rendering of an entire scene. A rule of thumb in surface rendering states that the quality of rendering is proportional to the number of polygonal elements. Unfortunately, the screen refresh rate of a graphics display is inversally proportional to the number of elements.
Therefore, an important concern arises when specifying a soft tissue model : is it compatible with high quality visual rendering ? For some models, it is clearly not the case. For instance the chain-mail algorithm [43] represents soft tissue with the help of cubic lattices that are allowed to move slightly with respect to their neighbors. For this representation, as well as for particle-based representations [40, 33] and multigrid representations [28], authors use a two-layer strategy : a volumetric soft tissue model is combined with a surface model dedicated to visual rendering. These two models are often coupled with a linear relationship based on barycentric coordinates : once the shape of a soft tissue model is modified, the surface model is updated is an efficient manner. Similarly, the collision detection is performed on the surface model, but contact forces and displacements are imposed on the volumetric model. However, this approach has two limitations. First, the modeling of contact between a virtual tool and a soft tissue model is usually not satisfactory because the mapping between surface and volumetric model is complex (though mapping from volumetric to surface models is often trivial). Second, this approach makes the modeling of tissue cutting very complex where the surface and volumetric topology is altered.
For soft tissue models based on tetrahedral or hexahedral meshes, the problem of high quality visual rendering is posed in a different manner since the shell of these meshes (made of triangular or quadrangular elements) can be used direcly for rendering. However, in general, coarse volumetric meshes are used in order to achieve real-time performances (see next section). Therefore, it is often required to compensate the poor geometrical quality by using specific computer graphics algorithms such as subdivision surfaces [112], using avatars [29] or by replacing elements with texture [98]. In the case of the hepatic surgery simulator, we have used the PN triangles algorithms [108] in order to provide a smooth visual rendering of the liver. The idea behind PN triangles is to subdivide each triangle and its normal vector into subtriangles in order to produce a smoother looking surface (see figure 4 for an example).

1.3.2 Real Time Deformation Constraints

A surgical simulator is an example of a virtual reality system. To succeed in training surgeons, a simulator must provide an advanced user interface that leads to the immersion of surgeons into the virtual surgical field. To reach this level of interaction, three basic rules must be formulated :

Rule 1 **Minimum bandwidth for visual and haptic feedback** An acceptable bandwidth for visual display is in the range of 20-60Hz while the acceptable bandwidth for haptic display is on the range of 300-1000Hz (300Hz is the free hand gesture frequency). In fact, this notion of minimal bandwidth



(a) (b) (c)

Figure 4: Display of a liver being resected : (a) Display of the triangles corresponding to the shell of the liver tetrahedral mesh; (b) Surface Rendering based on Gouraud Shading without PN triangles; (c) Surface Rendering based on PN triangles with two levels of subdivisions.

depends on the nature of the scene to be displayed : for objects moving slowly on the screen, an update rate of 20 Hz is sufficient. Similarly, a frequency of 300 Hz may be enough to render the contact with very soft objects.

Rule 2 **Low latency** Latency measures the time between measurements performed by the sensor (for instance the position of the surgical instrument) and action (visual or haptic display). Latency is critical for user immersion. The hardware configuration of the system can greatly influence latency since communication between elements may be responsible for additional delays. In figure 5, the architecture of the simulation system used at INRIA [24] in 1996 is presented. It is composed of one haptic display, a Personal Computer and a graphics workstation. There are several causes contributing to latency : communication delays between the haptic display and the PC, communication between the PC and the graphics workstation, the delay caused by the graphics display, the computation time for collision detection, force feedback and deformation. Since some of the communication links between elements are asynchronous, the total latency is not the sum of those delays but it is important to reduce them to their minimum values. The latency depends greatly on hardware, specifically on computation and graphics performance.

Rule 3 **Realistic motion of soft tissue** It is important that the dynamic behavior of a deformable tissue is correctly simulated. To assess the visco-elastic behavior of a material, one can measure the speed at which an object returns to its rest position after it is perturbed. Soft tissue undergoes a damped motion whereas stiff objects react almost instantaneously to any perturbation. At the limit, very stiff objects can be considered to have a quasi-static motion, implying that static equilibrium is reached at each time-step (see section 5 for a discussion about quasi-static motion).

In terms of soft tissue modeling, two parameters are important for real-time deformation constraints. The first parameter is the *update frequency* f_u which controls the rate at which the shape of a soft tissue model is modified. If we write X_t as the position of the tissue model at iteration t, the *computation time* $T_c = \frac{1}{f_u}$ is the time needed to compute the new position X_{t+1}. The

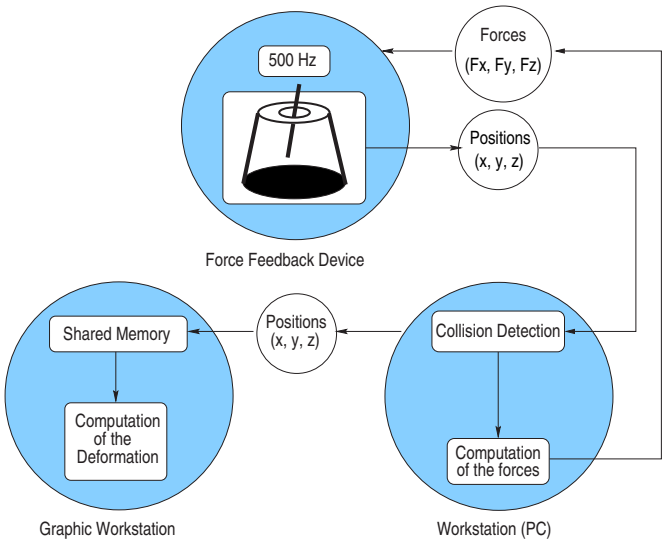


Figure 5: Architecture of a simulator composed of a Personal Computer driving an haptic device and a graphics workstation.

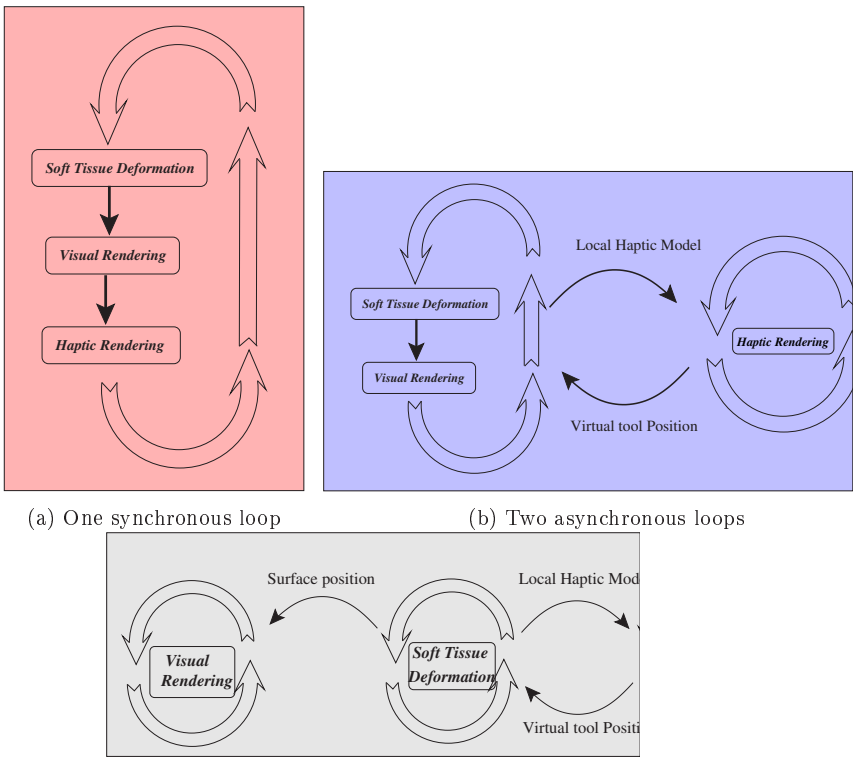
second parameter is the *relaxation time* $T_{relaxation}$ which is the time needed for a material to return to its rest position once it has been perturbed.

To reach the required bandwith for haptic and visual rendering (rule number 1) it is necessary that the computation time T_c is bounded by a constant $T_{interaction}$ that depends on the architecture of the simulator. For instance, in figure 6 we display three different software architectures for handling soft tissue deformation, visual and haptic feedback.

In a first architecture (Figure 6 (a)), all three tasks are performed sequencially, one after the other. The advantage of this approach lies in its simplicity of implementation. However, it has two drawbacks. The main problem is that the computation time T_c must be short enough to follow the minimum frequency for haptic feedback : 300 to 1000 Hz. This implies that $T_{interaction} \approx 2ms$ which is a very high requirement for a soft tissue model of reasonable size. In fact, to the best of our knowledge, only methods based on pre-computation of the static solution such as the one proposed in section 5 can comply with this constraint. The second problem with this approach is that a delay in any of the three tasks perturbs the other tasks. For instance, when the user performs tissue cutting, an additional task is needed to update the mesh topology which translates into a delay in the visual and haptic rendering.

The second architecture shown in Figure 6 (b) is the most commonly used in today's surgical simulators : the haptic rendering is performed in a different process or different thread than the visual rendering and soft tissue modeling tasks. Its purpose is to sharply decrease the real-time constraint on the soft tissue computation from haptic frequency ($\approx 500Hz$, $T_{interaction} \approx 2ms$) to visual frequency ($\approx 25Hz$, $T_{interaction} \approx 40ms$). In order to keep a satisfactory force feedback, a separate thread or process, running at haptic frequency, computes the force intensity for the haptic device based on a simplified local model. This

local model, that may consist of a sphere [73] or a plane [37] is updated by the soft tissue modeling loop while the position of the virtual surgical tool, necessary to compute its contact with soft tissue, is updated by the haptic rendering process and sent to the process. This asynchronous communication between haptic and visual rendering gives satisfactory results when some temporal smoothing is performed during the computation of force intensity. The main drawback of this approach is that it increases software complexity compared to the previous architecture. However, since only little information must be shared between the two processes, it has been adopted in several simulators, including the current version of the INRIA hepatic surgery simulator.



(a) One synchronous loop

(b) Two asynchronous loops

(c) Three asynchronous loops

Figure 6: Different software architecture for handling visual rendering, haptic rendering and soft tissue modeling.

In the third architecture described in Figure 6 (c), the visual and haptic rendering tasks are performed in separate processes or threads in order to remove the latency caused by graphics hardware. Furthermore, this architecture makes the computation of soft tissue deformation more efficient (decrease of T_c) when compared to previous solutions. However, it has little effect on the maximum computation time per iteration $T_{interaction} \approx 40ms$ since the geometric model still requires to be updated at 25 Hz for visual rendering. This approach is more difficult to implement because the amount of information to transmit to the visual rendering task is quite large. Furthermore, a change in mesh topology during simulation requires to modify the data structure of the computer

graphics algorithm. An example of such architecture is provided in [10].
To summarize, we can state that a soft tissue model in a surgical simulator must essentially meet two constraints : computation time T_c per iteration less than a constant $T_{interaction}$, and relaxation time $T_{relaxation}$ defined by the visco-elastic behavior of the material.

1.3.3 Tissue Cutting and Suturing

The ability to cut and suture tissue is of primary importance for designing a surgery simulation system. The impact of such operations in terms of tissue modeling is considerable since it implies changing tissue topology over time. The cost of such a topological change depends largely on the chosen geometric representation but also on the numerical method that is adopted to compute tissue deformation (see discussion in previous section).
In addition, the tissue behavior must be adapted at locations where cutting or suturing occurs. Little is known about the stress/strain relationship occuring during and after cutting. The basic assumption that is made is that the physical properties of tissue are only modified locally. However, in practice, cutting can modify the boundary conditions significantly between tissue and the surrounding organs which implies considerable change with respect to their ability to deform.

Finally, when cutting volumetric or surface models, it is very likely that the new geometric and physical representation of tissue leads to self-intersections. The detection of self-intersections is very computationally expensive and, therefore repulsive force between neighboring vertices are sometimes added to prevent self-intersections.

1.4 Computational Methods for Soft-tissue Modeling

Several computational methods can be employed for modeling the deformation of soft tissue. We simplify the taxonomy of these methods by proposing the three classes of algorithms most commonly used :

- **Direct Methods** This category contains all methods that solve the static or dynamic equilibrium equation at each iteration (quasi-static motion). To reach such performance, some kind of pre-computation is performed. The algorithm presented in section 5 is a direct method as well as the algorithm described in [28, 90]

- **Explicit Iterative Methods**. With iterative methods, the deformation is computed as the limit (in finite time) of a converging series that have been initialized. The closer the initial value is from the solution the faster the convergence. Iterative methods can be implemented based on implicit or explicit schemes. With explicit schemes, the next position of the tissue model X_{t+1}, is obtained from the application of internal forces estimated at iteration t. These methods encompass the most common algorithms found in the literature for modling soft tissue deformation, including spring-mass model [60], Tensor-Mass models [23] (presented in section 6), the "chain-mail" algorithm [43] and others [12].

- **Semi-Implicit Iterative Methods** With implicit or semi-implicit schemes, the next position of the tissue model X_{t+1}, is obtained from the application of internal forces estimated at iteration $t+1$. Therefore, a linear system of equations needs to be solved entirely or partially [4].

In table 1, we present the general features of these three types of numerical methods with respect to the constraints enumerated in section 1.3. More precisely the time of computation, the relaxation time (inversely proportional to the speed of convergence towards the rest position) and the ability to support any change of mesh topology during the simulation of cutting or suturing is estimated qualitatively for each method.
Direct methods can support high frequency update f_u and may have a low relaxation time to model stiff material : but they cannot simulate tissue cutting since they rely on the precomputation of some parameters.

	Direct Methods	Explicit Iterative Scheme	Implicit Iterative Scheme
Computation time	low	low	high
Relaxation Time	low	high	low
Cutting simulation	Very difficult	possible	difficult

Table 1: Comparison between the three soft tissue models: direct methods (pre-computed quasi-static model), explicit iterative schemes (Tensor-Mass and spring-mass models) and implicit iterative schemes (Houbolt or Newmark methods).

On the other hand, explicit iterative methods are well suited for the simulation of cutting, but these method often suffer from a high relaxation time, which makes their dynamic behavior somewhat unrealistic (jelly-like behavior). This high relaxation time originates from a lack of synchronicity, where the time step Δt used in the discretization of the explicit scheme, is much smaller than the computation time T_c. To obtain satisfactory results, it is often required to use a mesh with a small number of nodes (typically less than 1000 vertices on a standard PC).
Finally, with implicit iterative methods, the time step Δt can be increased by an order of magnitude compared to the explicit case. This allows to obtain much better dynamical behavior, but, on the other hand, they suffer from a higher computation time than explicit methods since a (sparse) linear system of equations needs to be solved at each iteration. Again, to achieve real-time performance, these methods are limited to meshes with a small number of vertices.

2 The INRIA Hepatic Surgery Simulator

2.1 Objectives

In the sequel we use the hepatic surgery simulator developped at INRIA in the Epidaure project[2] as a case study to illustrate the different algorithms and the practical issues involved when building soft tissue models.

[2]Description of the Epidaure project is provided at http://www.inria.fr/epidaure/

The INRIA Hepatic Surgery Simulator was initiated in 1995 as a part of the European project MASTER in collaboration with the IRCAD research center[3] which hosts the European Institute of Tele-Surgery (EITS). The motivations that led us to propose the development of an hepatic surgery simulator were twofold.
First, hepatic pathologies are among the major causes of death worldwide. For instance, hepatocellular carcinoma (HCC) is a primary liver cell cancer and it accounts for most of cancer tumors. It causes the death of 1 250 000 people mainly in Asia and Africa. Furthermore, hepatic metastases (secondary tumorous cells) are mainly caused by colorectal cancers (in 30% to 50% of cases) and patients have little chance to survive hepatic carcinoma without any therapy (0 to 3% of survival for a 5 year period with an average survival period of 10 months).
The second motivation is related to the nature of hepatic resection surgery. Indeed, this surgical procedure involves many generic surgical gestures (large displacement motion, grasping, cutting, suturing) that can be useful in the simulation of different procedures. Also, because of the presence of hepatic parenchyma, the tissue models must be of volumetric nature which departs significantly from previously developped simulators simulating hollow organs like the gall bladder. Finally, tissue being a soft material allows to employ low-end force feedback systems for simulating contact forces between surgical tools and hepatic tissue.
This work has greatly benefited from the INRIA incentive action AISIM[4] which gathered different INRIA teams working in the fields of medical image analysis (Epidaure), robotics (Sharp) [11], computer graphics (Imagis) [28] and numerical analysis (Sinus, Macs) [107].

2.2 Liver Anatomy

The liver is the largest gland (average length of about 28 cm, average height of about 16 cm and average greatest thickness of about 8 cm) in the human body. It has numerous physiological functions: to filter, metabolize, recycle, detoxify, produce, store and destroy. It is located in the right hypochondriac and epigastric regions. The liver has a fibrous coat, the so-called Glisson's capsule. Its rheological behavior is quite different from the glandular parenchyma. Five vessel types run through the liver parenchyma: biliary and lymphatic ducts on one hand, blood vessels (internal portal supply, hepatic arterial tree and collecting venous network) on the other hand. The portal vein, which conveys blood from the digestive tract to be detoxified and metabolized, is deep to the proper hepatic artery and common bile duct. This hepatic triad runs to the liver; it enters the liver via the hilum. This region is thus supposed to be wholly stable.

2.3 Creation of an anatomical model of the liver

In order to produce a model of the liver with anatomical details, the Visible Human dataset [1] provided by the *National Library of Medicine* was used. This

[3] Institut de Recherche Contre le Cancer de l'Appareil Digestif, 1, Place de l'Hôpital, 67091 STRASBOURG Cedex, http://www.ircad.com/, funded by Pr. J. Marescaux
[4] http://www-sop.inria.fr/epidaure/AISIM/

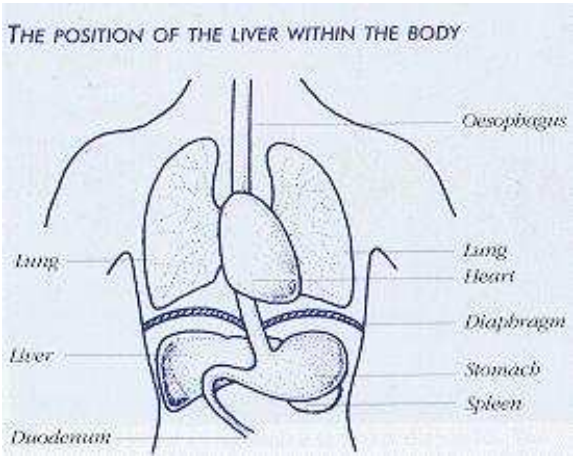


Figure 7: Description of the liver anatomy with its neighboring structures (source Children's Liver Disease Foundation)

dataset consists of axial MRI images of the head and neck and longitudinal sections of the rest of the body. The CT data consists of axial scans of the entire body taken at 1 mm intervals. The axial anatomical images are scanned pictures of cryogenic slices of the body. They are 24-bit color images whose size is 2048 by 1216 pixels. These anatomical slices are also at 1 mm interval and are registered with the CT axial images. There are 1878 cross-sections for each modality.

To extract the shape of the liver from this dataset, we used the anatomical slices (cf. figure 8), which give a better contrast between the liver and the surrounding organs. The dataset corresponding to the liver is reduced to about 180 slices. After contrast enhancement, we apply an edge detection algorithm to extract the contours of the image, and then using a simple thresholding technique, we retain only those with higher-strength contours are considered for further processing. Next, we use semi-automatic deformable contour [55, 32] to extract a smooth two-dimensional boundary of each liver slice. These contours are finally transformed into a set of twodimensional binary images (cf. figure 8). The slices generated are then stacked to form a tridimensional binary image [75] (cf. Figure 9).

In order to capture the shape of the external surface of the liver, one could use a sub-voxel triangulation provided by the marching-cubes algorithm [65], however the number of triangles generated is too large for further processing. Moreover, a smoothing of the surface is necessary to avoid staircase effects (see figure 9). A possible solution consists in decimating an iso-surface model by using a mesh simplification tool [95]. However, for more flexibility, in both the segmentation and simplification processes, liver reconstruction was performed using *simplex meshes*.

Simplex meshes are an original representation of tridimensional objects developed by Delingette [31]. A simplex mesh is a deformable discrete surface mesh that is well suited for generating geometric models from volumetric data. A simplex mesh can be deformed under the action of regularizing and external forces. Additional properties like a constant connectivity between vertices and

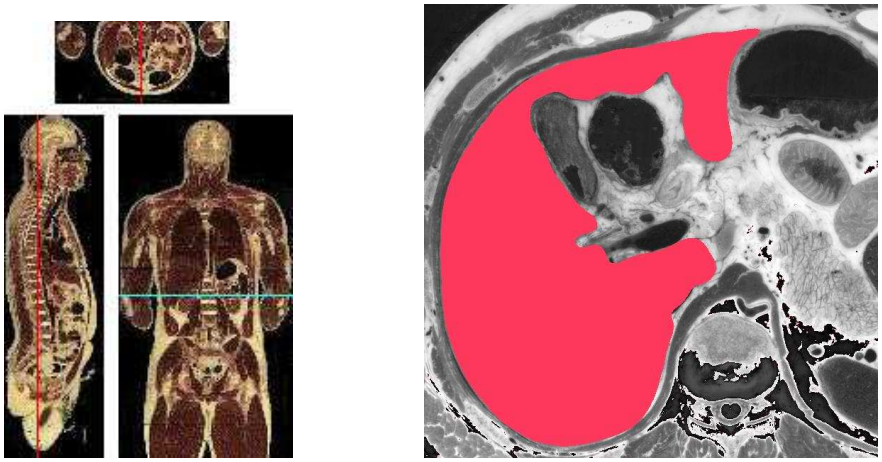


Figure 8: Slice-by-slice segmentation of the liver. The initial data (left) is a high resolution photography of an anatomical slice of the abdomen. The binary image (right) corresponds to the segmented liver cross-section.

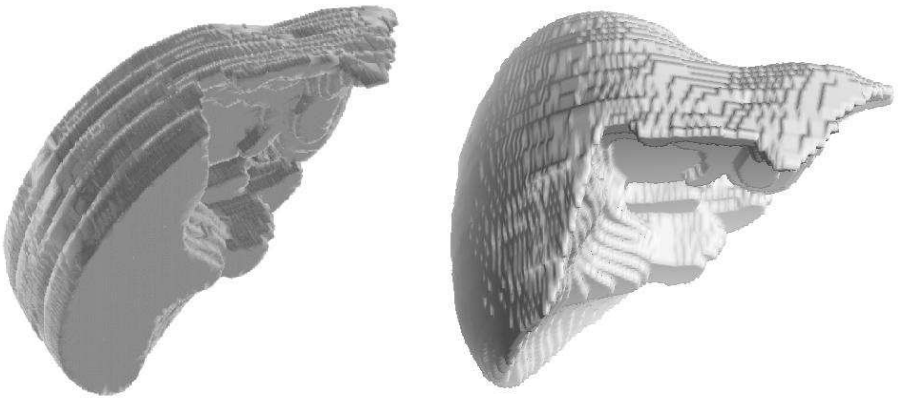


Figure 9: After segmentation, the binary images are stacked (left) to give a 3D binary image. We see the step-effect on the shape of the liver (right) when extracted using the marching-cubes algorithm.

a duality with triangulations have been defined. Moreover, simplex meshes are adaptive, for example by concentrating vertices in areas of high curvature (thereby achieving an optimal shape description for a given number of vertices). The mesh may be refined or decimated depending on the distance of the vertices from the dataset. The decimation can also be interactively controlled. Figure 10 shows the effect of the mesh adaptation and where the vertices are nicely concentrated at highly high curvature locations of the liver.

By integrating simplex meshes in the segmentation process, we have obtained smoothed triangulated surfaces, very close to an iso-surface extraction, but with fewer faces to represent the shape of the organs. In our example, the liver model has been created by fitting a simplex mesh to the tridimensional binary image previously described. Thanks to the adaptation and decimation properties of the simplex meshes, this model is composed of only 14,000 triangles, whereas the marching-cubes algorithm produced 94,000 triangles (cf. Figures 9 and 10).

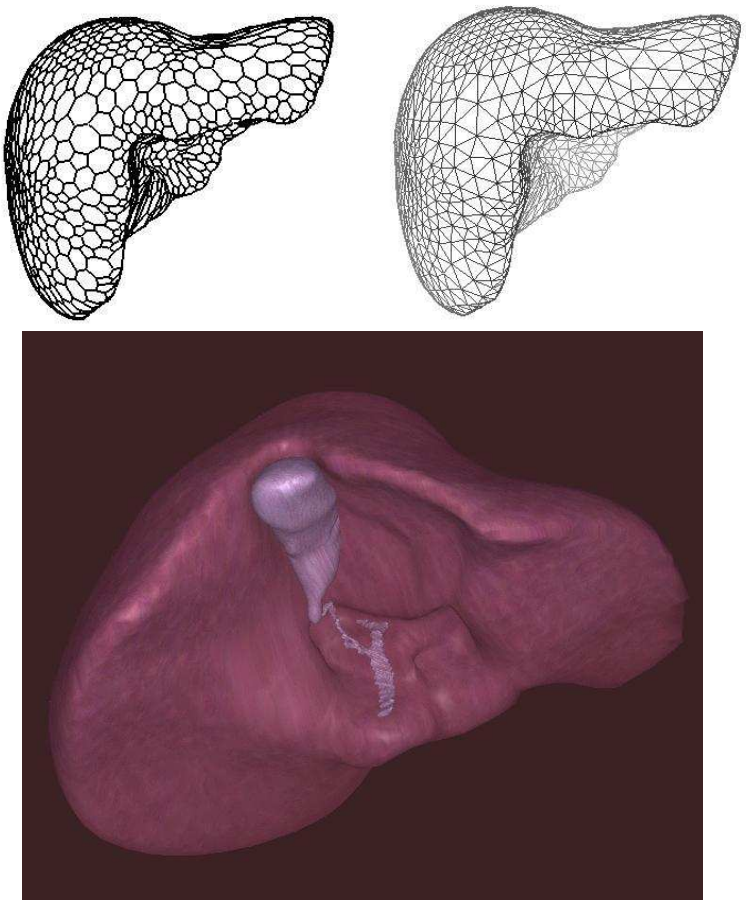


Figure 10: Different representations of the geometric liver model. The simplex mesh [75] fitting the data (top left) with a concentration of vertices in areas of high curvature, the triangulated dual surface (top right) and a texture-mapped model with anatomical details (gall bladder and ducts) from an endoscopic viewpoint (bottom).

Although this approach is very useful for building a "generic" liver model, it is essential to integrate "patient-based" models in the simulator. In the framework of this research project, Montagnat and Delingette [75] have developped a method for extracting liver models from CT scan images. The principle of this algorithm is to deform a generic simplex mesh (for instance the one extracted from the Visible Human dataset) such that its surface coincides with the liver boundary in the image. The work of Soler *et al.* [102, 101] has extended this work by additionnally extracting the main bifurcations of the portal and hepatic veins but also the hepatic lesions and gall-bladder (see figure 11).

2.4 Liver Boundary Conditions

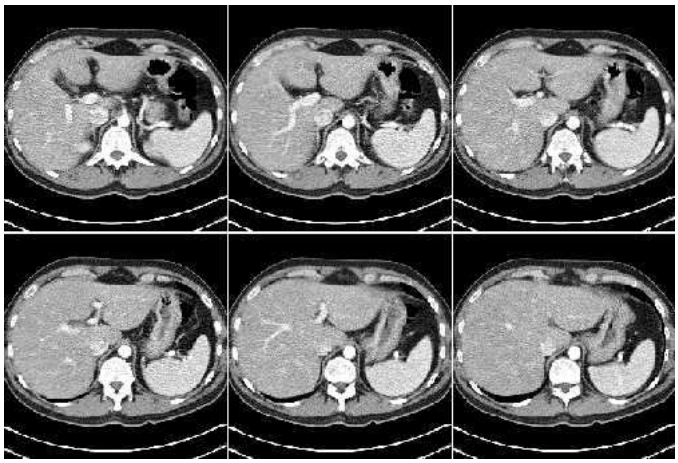
In the scope of the AISIM project, a reference liver model was created by M. Thiriet and M. Vidrascu [107]. They define the liver environment [107] in order to set up the boundary conditions associated to computational models. The right liver extremity is thick and rounded while the left one is thin and flattened. Both extremities are not submitted to specific loads. The anterior border is thin, sharp and free. The posterior border is connected to the diaphragm by the coronary ligament. The upper surface, covered by the peritoneum, is divided into 2 parts by the suspensory ligament. However, this ligament does not affect the biomechanical behavior of the liver. The lower surface is connected with the gall-blader (GB) within the GB fossa, the stomach, the duodenum,the right kidney and the right part of the transverse colon. These organs are in contact with the liver surface, but they do not interact strongly with the liver; they can not be considered as being supporting organs. The inferior vena cava (IVC) travels along the posterior surface, very often in a groove. The connection implies another strong fitting condition (clamp).

2.5 Material Characteristics

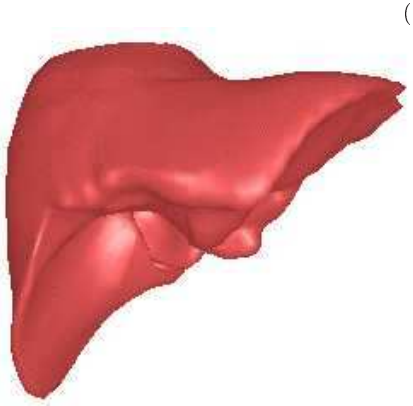
The literature on the mechanical property of the liver is relatively poor, but during the past four years, there has been a renewed attention on soft tissue characterization due to the development of new robotics tools and new imaging modalities. The published materials concerning liver biomechanical properties usually include two distinct stages. In a first stage, experimental curves relating strain and stress are obtained from specific experimental setups and in a second stage, parameters of a known constitutive law are fitted to these curves. Concerning the first stage, there are three different sources of rheological data :

- **ex-vivo testing** where a sample of a liver is positioned inside a testing rig,
- **in-vivo testing** where a specific force and position sensing device is introduced inside the abdomen to perform indentation,
- **image-based elastometry** where an imaging modality like ultrasound [111], Magnetic Resonance Elastometry [69] or CT-scan [78, 46] provides relevant information to assess the Young Modulus of living materials.

A non-comprehensive list of articles describing the liver material characteristics is provided in table 2. From this wide variety of studies, it is difficult to pick one

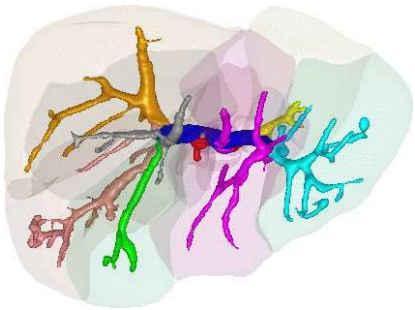


(a)

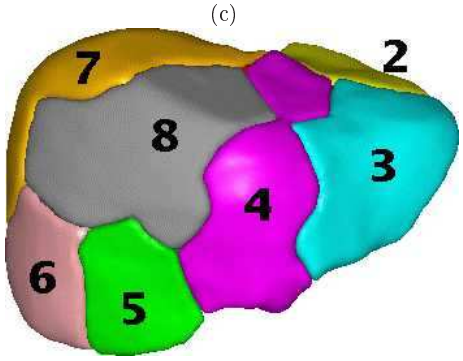


(b)

(c)



(d)



(e)

Figure 11: (a) Original CT-scan images of the liver; (b) reconstructed liver model; (c) outline of the liver surface model in a CT-scan image; (d) Segmentation of the portal vein [101]; (e) reconstruction of the eight anatomical segments (Couinaud segmentation).

particular constitutive model since each of experimental setup has its advantages and drawbacks. For instance, the rich perfusion of the liver affects deeply its rheology (the liver receives one fifth of the total blood flow at any time) and therefore it is still an open question whether *ex-vivo* experiments can assess the property of living liver tissue, even when specific care is taken to prevent the swelling or drying of the tissue. Conversely, data obtained from *in-vivo* experiments should also be considered with caution because the response may be location-dependent (linked to specific boundary conditions or non-homogeneity of the material) and the influence of the loading tool caliper on the deformation may not be well understood. Furthermore, both the respiratory and circulatory motions may affect *in-vivo* data.

First Author	Experimental Technique	Liver Origin	Young Modulus (kPa)
Yamashita [111]	Image-Based	Human	Not Available
Brown [15]	*in-vivo*	Porcine Liver	≈ 80
Carter [17]	*in-vivo*	Human Liver	≈ 170
Dan [27]	*ex-vivo*	Porcine Liver	≈ 10
Liu [62, 61]	*ex-vivo*	Bovine Liver	Not Available
Nava [76]	*in-vivo*	Porcine Liver	≈ 90
Miller [74]	*in-vivo*	Porcine Liver	Not Available
Sakuma [92]	*ex-vivo*	Bovine Liver	Not Available

Table 2: List of published articles providing some quantitative data about the biomechanical properties of the liver.

Furthermore, little is known about the variability of liver characteristics between species (does a porcine liver behave like a human liver ?) but also between patients. For instance, studies from Nava *et al.* [76] suggest that a 20% difference in stiffness between normal and diseased livers whereas Brown [15] *et al.* show significant differences between *in-vivo* pig livers and *ex-vivo* cow livers.

Another important source of uncertainty in those measurements is the strain state of the liver during indentation. Indeed, as pointed out by Brown [15], most researchers precondition their liver samples by applying several cycles of indentation in order to have more consistent estimates of stiffness and hysteresis. However, during surgery, (rightfully) surgeons do not precondition living tissues which may imply that only measurements obtained *in-vivo* and *in-situ* through modified surgical instruments (like those developed in [17, 15, 76]) are relevant for modeling soft tissue in a surgical simulator.

Finally, the rheology of the liver is not only influenced by its perfusion, but also by the Glisson's capsule. For instance, Carter [17] *et al.* have showed that the stiffness of cylindrical samples of liver parenchyma with part of Glisson's capsule is twice the one without Glisson's capsule, using similar rheological tests [17].

To conclude, more experimental studies are needed to reach a good understanding of the liver biomechanical properties. Methods based on *in-vivo* and *in-situ* indentations seem to be the most promising ones for building realistic soft tissue models in surgery simulation. All studies demonstrate that the liver is a strongly visco-elastic material, while Liu [62] *et al.* suggest that the liver can be considered as linear elastic for strains smaller than 0.2 %.

Fortunately, in many surgical simulators, the boundary conditions governing the deformation of soft tissues, consist of imposed displacements only. In such case, the computation of soft tissue deformation requires to solve an homogeneous system of equations $\mathbf{FU} = \mathbf{0}$ which is not sensitive to the absolute value of stiffness materials but to the relative stiffness between materials [44]. Hopefully, we can expect that the relative stiffness between the liver and its neighboring organs is less variable and easier to assess, for instance through medical imagery.

3 Linear Elastic Models for surgery Simulation

3.1 Main features of our approach

In the next sections, we propose three different soft tissue models that are well-suited for the simulation of surgery and which are compatible with the constraints described in section 1.3. These models bear many common features that are listed below :

- Volumetric Structures
- Continuum mechanics based deformation
- Finite Element Modeling
- Linear Tetrahedron Finite Element
- Strong Approximation in Dynamical Modeling

We explain the motivations of such characteristics in the next sections.

3.1.1 Using Volumetric Models

We can classify the geometry of anatomical structures depending on their "idealized" dimensionality, even though they all consist of an assembly of tridimensional cells. For instance, at a coarse scale, a blood vessel can be thought as a one-dimensional structure [89] whereas the gall-blader can be represented as a two-dimensional structure [60] (a closed surface filled with bile). Similarly, the behavior of most parenchymatous organs such as the brain, lungs, liver or kidneys are intrinsically volumetric. But one should notice that at a fine enough scale, all anatomical structures can be considered as volumetric.
In surgical simulators, it is frequent to rely on such dimensionality simplification in order to speed-up computation : tubular surfaces, such as the colon, are modeled as a deformable spline [39] whereas deformable volumetric structures, such as the liver, are represented with their surrounding surface envelope [60]. However, such artifices cannot be used in a hepatic resection simulator when the removal of hepatic parenchyma is performed.

3.1.2 Using Continuum mechanics

We have chosen to rely on the theory of continuum mechanics to govern the deformation of our volumetric soft tissue models. Other alternative representations exist such as spring-mass models [60], chain-mail [43] or long element models [20]. Spring-mass models correspond to small deformation one-dimensional

elastic elements (see section 6.1.7 for an extended comparison) but are no longer valid for two or three-dimensional elasticity. These models are especially popular in computer graphics since they are easy to implement and are based on straightforward point mechanics. The chain-mail [43] is an original quasi-static deformable model based on a hexahedral mesh which is well suited for stiff material but does not allow any topological change. Long element models [20] correspond to valid tridimensional cylindrical elastic models but are used to approximate the deformation of general volumetric shapes.

We chose to base our soft tissue models on continuum mechanics since it offers a well-studied and validated framework for modeling the deformation of volumetric objects unlike the methods cited above. Furthermore, it offers the following advantages :

- **Scalability** : when modifying the mesh topology (refinement or cutting for instance), the behavior of the mesh is guaranteed to evolve continuously.

- **Physical Parameter Identification**: the elastic parameters of a biomaterial (Young modulus for instance) can be estimated from various methods (incremental rheological experiments, elastography or solving inverse problems). Parameter identication for spring-mass models is known to be more difficult and requires stochastic optimization (genetic algorithms [66] or simulated annealing [34]).

3.1.3 Using Finite Element Modeling

Finite Element Modeling (FEM) is certainly the most popular technique for the computation of structure deformation based on the elasticity theory. Furthermore, it is well formalized and understood and there exists many software implementations although none of them deals with real-time deformation. Nonetheless there exists alternative approaches such as Boundary Element Modeling (BEM) or the Finite Difference Method (FDM).

The BEM is well suited for the simulation of linear elastic isotropic and homogeneous materials (for which there exists a Green function) and is indeed a good alternative to FEM when the mesh topology is not modified. In fact BEM has the important advantage over FEM of not requiring the construction of a volumetric mesh. A more thorough discussion is provided in section 5.3.2 but this approach is not well suited when cutting is simulated.

The FDM is well-suited when the domain is discretized over a structured grid in which case partial derivatives can be easily discretized. They often lead to the same equation as FEM when specific finite elements (based on linear interpolation) are employed [6]. On unstructured meshes such as tetrahedral meshes, some extensions of the finite difference method have been proposed [28] also leading to a similar equation as FEM (see discussion in section 4.4). With non-linear elasticity however, FEM [82] and FDM [28] differ significantly and no formal proof has been given that the FDM converges towards the right solution as the mesh resolution increases.

3.1.4 Using Linear Tetrahedron Finite Element

For all finite element models described in the remainder, a simple finite element is used : a 4-node tetrahedron with linear interpolation ($P1$). The Linear Tetrahedron (LT) is known to be a poor element (in terms of convergence) compared to the Linear Hexahedron (LH) for static linear and non-linear elastic analysis [8]. Also this paper shows that LH performs better than the Quadratic Tetrahedron (10 nodes) even in a static linear elastic analysis.
The motivation for using tetrahedra rather than hexahedra clearly comes from a geometrical point of view. Indeed, meshing most anatomical structures with hexahedra is known to be a difficult task especially for structures having highly curved or circumvoluted parts such as the liver or the brain parenchyma. To obtain a smooth surface envelope, it is then necessary to employ many hexahedra where a smaller number of tetrahedra would suffice. Furthermore, there exist several efficient commercial and academic software [99, 79] to fill automatically a closed triangulated surface with tetrahedra of high shape quality [81]. A second motivation for using tetrahedra rather than hexahedra is related to the simulation of cutting soft tissue that involves removing and remeshing of local elements. With hexahedral meshes, it is not possible to simulate general surface of cut without resorting to add new element types (such as prismatic elements). Such multi-element models [6] would make the matrix assembly and local remeshing algorithms more complex to manage.

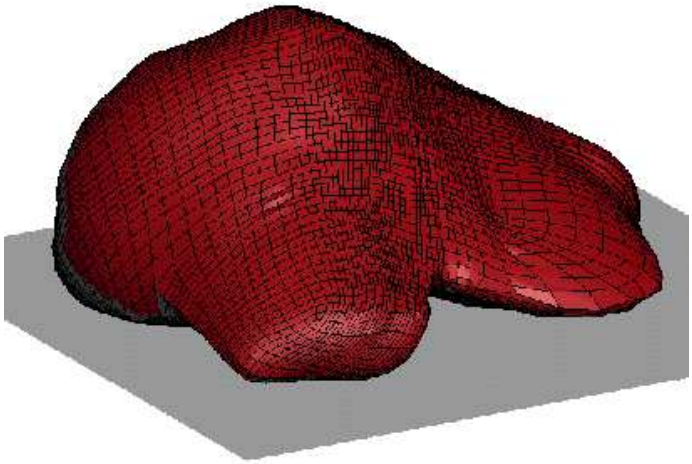


Figure 12: Example of liver meshed with hexahedra (courtesy of ESI SA)

Regarding the choice of the interpolation function (linear versus quadratic), our choice has been mainly governed by computational issues. Given that a minimum number of tetrahedra is necessary to get a realistic visual rendering of a structure, the QT element involves one additional node per edge compared to the LT element which on a typical tetrahedral mesh implies at least a six-fold increase of the number of nodes. Furthermore, we believe that the loss of

accuracy in the deformation computation entailed by the use of LT elements remains small compared to the large uncertainty on the physical parameter values (Young modulus,..) existing for most soft tissues.
Finally, by using linear elements, the computation of local stiffness matrices can be done explicitely (analytically) even for non-linear elasticity. Also, the gradient of the displacement field which is constant inside each element (constant strain) has a simple geometric interpretation using area vectors (see section 4.2). A significant speed-up is therefore obtained when compared to higher order elements that require numerical integration methods such as Gauss quadrature for estimating stiffness matrices.

3.1.5 Using large approximations of Dynamic Behavior

Despite the development of new *in vivo* rheological equipment [56], the dynamical behavior is only known quantitatively for a few anatomical structures : skin, muscle, myocardium, ... The viscoelastic properties of liver tissue have been studied by Liu *et al.* [61] but for most organs, constitutive laws of dynamics and their parameters must be hypothesized and validated qualitatively.
In a surgical simulator, the boundary conditions caused by the contact with surgical instruments can change between two iterations. Given that surgeons typically move their instruments at low speed (typically a few milimiters per second) and making the hypothesis that the mass of these instruments is the same or smaller than the mass of anatomical structures, we chose to neglect the dynamics of soft tissue models in two different ways. For a first class of models described in section 5, we solve the static problem $\mathbf{F} = \mathbf{KU}$ (where $\mathbf{F}$ is the force vector, $\mathbf{K}$ the stiffness matrix and $\mathbf{U}$ the displacement vector) at each iteration thus leading to a quasi-static approximation.
For a second class of models, described in sections 6.1 and 7, we solve the Newtonian equation of motion $\mathbf{M\ddot{U}} + \mathbf{C\dot{U}} = -\mathbf{KU}$ with the following hypotheses : the mass matrix $\mathbf{M}$ is proportional to the identity matrix while the damping matrix $\mathbf{C}$ is diagonal. Furthermore, in some cases, the computational time T_c is longer than the time step Δt which creates a lack of synchronicity in the simulation.

3.2 Tridimensional Linear Elasticity

The fast computation of soft tissue deformation in a surgical simulator requires that some hypotheses are made about the nature of the tissue material. A first hypothesis, which leads to the two soft tissue models described in section 5 and 6, assumes that soft tissue can be considered as linear elastic. The rationale behind this hypothesis is clear : the linear relation between applied forces and node displacements leads to very computationally efficient algorithms. But, linear elasticity is not only a convenient mathematical model for deformable structures : it is also a quite realistic hypothesis. Indeed, all hyperelastic materials can be approximated by linear elastic materials when small displacements (and therefore small deformations) are applied [41, 72]. It is often admitted as reasonable to consider that a material is linear elastic when observed displacements are less than 5 % of the typical object size. In the case of hepatic tissue, a recent publication [61] indicates that the linear domain is only valid for strain less than 0.2%.

Whether this constraint on the amount of displacement is valid or not in a surgical simulator depends both on the anatomical structure and the type of surgery. For instance, when simulating the removal of the gall bladder (cholelysectomy), the liver undertakes small displacements but it is not the case when simulating hepatic resection or any other surgical procedure that requires a large motion of the left lobe.
When large displacements are applied to a linear elastic material, the approximation of hyperelasticity is no longer valid and large errors in the computation of deformation and reaction forces can be perceived both visually and haptically. Section 7.1 describes the shortcomings of linear elasticity in such cases.
To summarize the general equations of linear elastic materials, we proceed in four steps. In section 3.2.1, we provide some general definitions whereas sections 3.2.3 and 3.2.4 give the main equations of isotropic and transversally anisotropic material. Finally, the principle of virtual work is formulated in section 3.2.5.

3.2.1 Definition of infinitesimal strain

We consider a three dimensional body defined in a tridimensional Euclidian space $\mathbb{R}^3$. We describe the geometry of this body in its rest position $\mathcal{M}_{rest}$ by using material coordinates $\mathbf{X} = (x, y, z)^T$ defined over the volume of space Ω occupied by $\mathcal{M}_{rest}$.

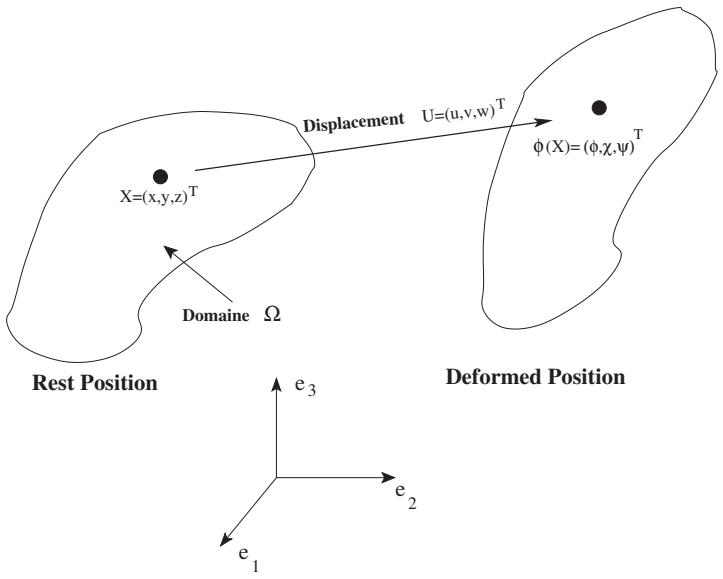


Figure 13: Definition of deformation and displacement between rest and deformed positions

This body is deformed under the application of boundary conditions : these may be either geometric boundary conditions (also called essential boundary conditions [6]) or natural boundary conditions *i.e.* prescribed boundary forces. We note $\mathcal{M}_{def}$ the body in its deformed state and $\mathbf{\Phi}(x, y, z)$, the deformation function that associates to each material point $\mathbf{X}$ located in the body at its rest

position, its new position $\mathbf{\Phi}(\mathbf{X})$ after the body has been deformed :

$$\begin{array}{rlll} \mathbf{\Phi}: & \Omega \subset \mathbb{R}^3 & \longmapsto & \mathbf{\Phi}(\Omega) \\ & \mathbf{X} & \longrightarrow & \mathbf{\Phi}(\mathbf{X}) = \left\{ \begin{array}{l} \phi(x,y,z) \\ \chi(x,y,z) \\ \psi(x,y,z) \end{array} \right. \end{array}$$

The displacement vector field $\mathbf{U}$ is defined as the variation between the deformed position and the rest position (see figure 13) :

$$\begin{array}{rlll} \mathbf{U}(\mathbf{X}): & \Omega & \longmapsto & \mathbb{R}^3 \\ & \mathbf{X} & \longrightarrow & \mathbf{U} = \left\{ \begin{array}{l} u(x,y,z) \\ v(x,y,z) \\ w(x,y,z) \end{array} \right. \end{array}$$

The observed deformation can be characterized and quantified through the analysis of the spatial derivatives of the deformation function $\mathbf{\Phi}(\mathbf{X})$. More precisely, the right Cauchy-Green strain tensor $\mathbf{C}(\mathbf{X})$ which is a symmetric 3×3 matrix (therefore has 3 real eigenvalues) is simply computed from the deformation gradient :

$$\mathbf{C}(\mathbf{X}) = \nabla \mathbf{\Phi}^T \nabla \mathbf{\Phi} \tag{1}$$

The Green Lagrange strain tensor $\mathbf{E}(\mathbf{X})$, derived from the right Cauchy-Green strain tensor, allows to analyze the deformation after rigid body motion has been removed :

$$\mathbf{E}(\mathbf{X}) = \frac{1}{2}(\mathbf{C} - \mathbf{I}_3) = \frac{1}{2}(\nabla \mathbf{U} + \nabla \mathbf{U}^T + \nabla \mathbf{U}^T \nabla \mathbf{U}) \tag{2}$$

where $\mathbf{I}_3$ is the 3×3 identity matrix.
In the the linear elasticity framework, applied displacements are considered as infinitesimal and the Green Lagrange strain tensor $\mathbf{E}(\mathbf{X})$ is linearized into the infinitesimal strain tensor $\mathbf{E}_{\mathrm{L}}(\mathbf{X})$. This symmetric 3×3 tensor is simply written as :

$$\mathbf{E}_{\mathrm{L}}(\mathbf{X}) = [e_{ij}] = \frac{1}{2}(\nabla \mathbf{U} + \nabla \mathbf{U}^T) = \begin{bmatrix} e_{xx} & e_{xy} & e_{xz} \\ e_{xy} & e_{yy} & e_{yz} \\ e_{xz} & e_{yz} & e_{zz} \end{bmatrix} \tag{3}$$

The diagonal elements e_{ii} of the symmetric matrix describe the relative stretch in the direction of the reference frame, whereas off-diagonal elements e_{ij} describe shearing quantities.

3.2.2 Definition of infinitesimal stress

The deformation of a tridimensional body is caused by applying external forces : these forces may be either body forces $\mathbf{F}^B$ (such as gravity forces) or surface forces $\mathbf{F}^S$ (applied pressure) or concentrated forces $\mathbf{F}^P$. As a reaction to external forces, internal forces are created inside the elastic body material.
Through Cauchy theorem [18], it is demonstrated that for each volume element inside the deformed body, the force per unit area $\mathbf{t}(\mathbf{X}, \mathbf{n})$ at a point $\mathbf{X}$ and along the normal direction $\mathbf{n}$ is written as :

$$\mathbf{t}(\mathbf{X}, \mathbf{n}) = \mathbf{T}(\mathbf{X})\mathbf{n}$$

where $\mathbf{T}(\mathbf{X})$ is the Cauchy stress tensor. The Cauchy stress tensor is a 3×3 symmetric tensor and can be written as :

$$\mathbf{\Sigma}(\mathbf{X}) = [\sigma_{ij}] = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_{zz} \end{bmatrix}$$

The Cauchy stress $\mathbf{\Sigma}$ and infinitesimal strain $\mathbf{E}_{\mathrm{L}}$ are conjugated variables [6] which implies the following relations :

$$\begin{aligned} \sigma_{ij} &= \frac{\partial W}{\partial e_{ij}} \\ e_{ij} &= \frac{\partial W}{\partial \sigma_{ij}} \end{aligned} \tag{4}$$

where $W(\mathbf{X})$ is the amount of elastic energy per unit volume.

3.2.3 Isotropic Linear Elastic Materials

For an isotropic linear elastic material, the elastic energy $W(\mathbf{X})$ is a quadratic function of the first two invariants of the infinitesimal strain tensor [18] :

$$W(\mathbf{X}) = \frac{\lambda}{2} \ (tr\mathbf{E}_{\mathrm{L}})^2 + \mu \ tr\mathbf{E}_{\mathrm{L}}^2 \tag{5}$$

where λ and μ are the two Lamé coefficients characterizing the material stiffness. These two parameters are simple functions of Young modulus E and Poisson coefficients ν, which belong to the material's physical properties :

$$\begin{aligned} \lambda &= \frac{E\nu}{(1+\nu)(1-2\nu)} & \mu &= \frac{E}{2(1+\nu)} \\ E &= \frac{\mu(3\lambda+2\mu)}{\lambda+\mu} & \nu &= \frac{\lambda}{2(\lambda+\mu)} \end{aligned}$$

Through equation 4, we can derive the linear relationship, known as Hooke's law, between the stress and the infinitesimal strain tensors :

$$\mathbf{\Sigma} = \lambda(tr\mathbf{E}_{\mathrm{L}})\mathbf{I}_3 + 2\mu\mathbf{E}_{\mathrm{L}} \tag{6}$$

Note that, the elastic energy can be written simply as a function of the linearized strain and stress tensors :

$$W = \frac{1}{2} \ tr(\mathbf{E}_{\mathrm{L}}\mathbf{\Sigma})$$

3.2.4 Transversally Anisotropic Linear Elastic Materials

Most anatomical structures like muscles, ligaments or blood vessels are strongly anisotropic material. This anisotropy is caused by the presence of different fibers (colagen, muscle, ...) that are wrapped together within the same tissue. For instance, anisotropic materials have been successfully used to model the deformation of the eye [54], of the heart [49, 48, 80] or the knee ligaments [109, 87]. In the scope of our hepatic surgery simulator, we have added an anisotropic

behavior where the first branches of the portal vein are located inside the hepatic parenchyma.
We have chosen to focus only transversally anisotropic material only where there is one direction $\mathbf{a}_0$ along which the material stiffness differs from the stiffness in the orthogonal plane. Indeed, one major obstacle when modeling anisotropic material is to get reliable data from rheological experiments regarding the directions of anisotropy and the Young modulus in all directions. With transversal anisotropy, it is sufficient to provide a single direction $\mathbf{a}_0$ and an additional pair of Lamé coefficients $\lambda^{\mathbf{a}_0}$ and $\mu^{\mathbf{a}_0}$ (see Figure 14).
The theoretical description of elastic energy of transversally anisotropic material is largely based on the work of Spencer [103, 104], and Fung [41]. For the sake of clarity, we introduce the notion of direction invariant and the concept of anisotropic stretching and shearing.

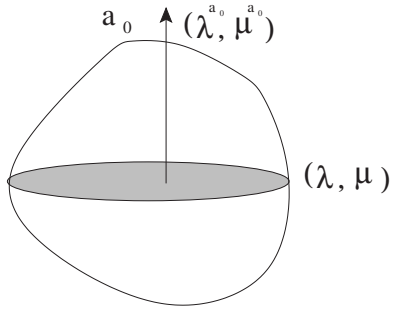


Figure 14: Definition of Lamé coefficients along the direction $\mathbf{a}_0$ are $\lambda^{\mathbf{a}_0}$ and $\mu^{\mathbf{a}_0}$.

We decompose the elastic energy of a transversally anisotropic material as the sum of the isotropic energy, provided by equation 5 and by a corrective term $\Delta W_{\mathrm{A}ni}$ which only depends on the variation of Lamé coefficients :

$$\begin{aligned} \Delta\lambda &= \lambda^{\mathbf{a}_0} - \lambda \\ \Delta\mu &= \mu^{\mathbf{a}_0} - \mu \\ W_{\mathrm{T}ransv.Ani}(\mathbf{X}) &= W(\mathbf{X}) + \Delta W_{\mathrm{A}ni}(\mathbf{X}, \Delta\lambda, \Delta\mu) \end{aligned}$$

Without loss of generality, we can assume that $\mathbf{a}_0$ coincides with the z direction of the reference frame. The isotropic elastic energy can then be written as a function of the stretch e_{zz} and shear (e_{xz}, e_{yz}) in the direction $\mathbf{a}_0$:

$$W(\mathbf{X}) = (\frac{\lambda}{2}+\mu)(e_{xx}^2+e_{yy}^2+e_{zz}^2)+\lambda(e_{xx}e_{yy}+e_{xx}e_{zz}+e_{yy}e_{zz})+2\mu(e_{xy}^2+e_{yz}^2+e_{xz}^2)$$

The purpose of the corrective term $\Delta W_{\mathrm{A}ni}$ is to modify the isotropic Lamé coefficients in the direction of anisotropy :

$$\Delta W_{\mathrm{A}ni}(\mathbf{X}) = (-\frac{\Delta\lambda}{2} + \Delta\mu)e_{zz}^2 + \Delta\lambda\ e_{zz}(\mathrm{tr}\mathbf{E}_{\mathrm{L}}) + 2\Delta\mu(e_{yz}^2 + e_{xz}^2)$$

The equation above can be written using the two parameters I_4 and I_5 which characterize the strain tensor in the direction $\mathbf{a}_0$ [84] :

$$I_4 = \mathbf{a}_0^T \mathbf{E}_{\mathrm{L}} \mathbf{a}_0 \quad (7)$$

$$I_5 = \mathbf{a}_0^T \mathbf{E}_{\mathrm{L}}^2 \mathbf{a}_0 \quad (8)$$

The first parameter I_4 is simply the amount of stretch in the direction $\mathbf{a}_0$ whereas the total amount of shearing $e_{xz}^2 + e_{yz}^2$ is given by $I_5 - I_4^2$. With these notations, the corrective term can be written as :

$$\Delta W_{\mathrm{A}ni}(\mathbf{X}) = \Delta\lambda \; I_4 \; tr\mathbf{E}_{\mathrm{L}} + 2\Delta\mu \; I_5 - (\frac{\Delta\lambda}{2} + \Delta\mu) I_4^2 \quad (9)$$

In [84], Picinbono *et al.* proposed to decompose the anisotropic term $\Delta W_{\mathrm{A}ni}(\mathbf{X})$ into a stretching and shearing part :

$$\begin{aligned} \Delta W_{\mathrm{A}ni}(\mathbf{X}) &= \Delta W_{Str.Ani} + \Delta W_{Sh.Ani} \\ \Delta W_{\mathrm{St}.Ani} &= (-\frac{\Delta\lambda}{2} + \Delta\mu) I_4^2 + \Delta\lambda \; I_4 \; tr\mathbf{E}_{\mathrm{L}} \\ \Delta W_{\mathrm{Sh}.Ani} &= 2 \; \Delta\mu (I_5 - I_4^2) \end{aligned}$$

In Figure 15, the distinction between stretching and shearing effects of a transversally anisotropic material is pictured by applying a force F_1 and F_2 on a cylinder respectively along and orthogonal to the direction.

3.2.5 Principle of Virtual Work

The equilibrium equation of a deformed body is derived through the *principle of virtual displacements*. This principle states that for any compatible virtual displacement $\hat{\mathbf{u}}(\mathbf{X})$ applied on a body $\mathcal{M}_{def}$, the total internal virtual work is equal to the total external work. The total internal work is given by the integral of elastic energy over the body volume whereas the external work is created by the application of body and surface forces :

$$\int_\Omega \hat{W}(\mathbf{X}) \; dV = \int_\Omega \hat{\mathbf{u}}^T \mathbf{f}^B \; dV + \int_{\partial\Omega} \hat{\mathbf{u}}^T \mathbf{f}^S \; dS \quad (10)$$

where $\mathbf{f}^B$ and $\mathbf{f}^S$ are the applied body and surface forces. Note that in equation 10, the virtual displacement field $\hat{\mathbf{u}}(\mathbf{X})$ is supposed to be compatible with the geometric boundary constraints (imposed displacements). Furthermore, this relation is only valid for small virtual displacements such that the linearized strain hypothesis still holds.

4 Finite Element Modeling

4.1 Linear Tetrahedron Element

As justified in section 3.1.4, the computation of soft tissue deformation is based on the finite element method. Anatomical structures of interest are spatially discretized into a conformal tetrahedral mesh. Conformality implies that the intersection of two tetrahedra of that mesh is either empty or consists of a vertex or an edge or a triangle.

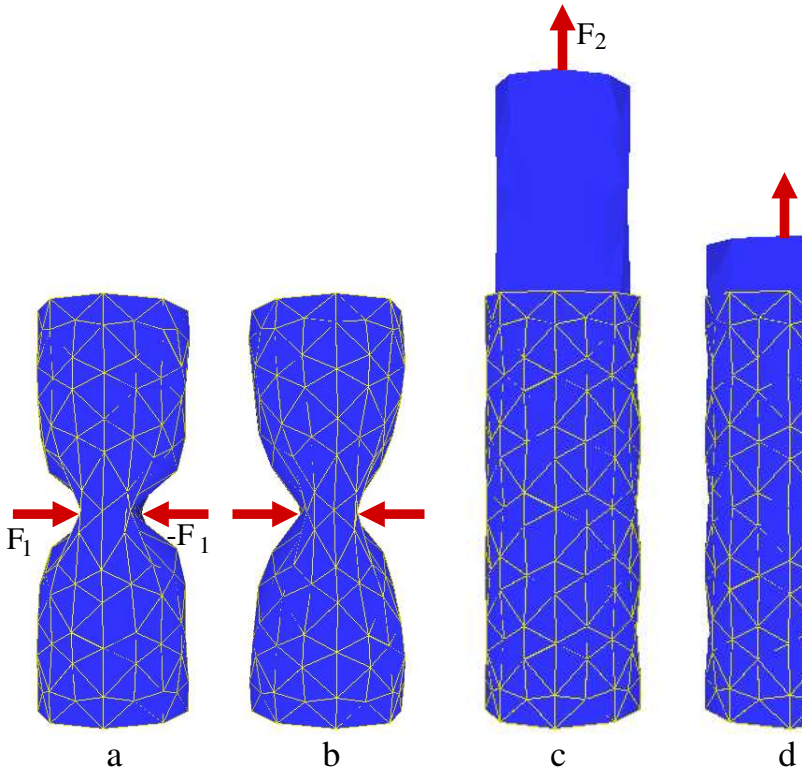


Figure 15: *Comparison between isotropic (a and c) and anisotropic (b and d) cylinders [82]. The same horizontal (resp. vertical) loads F_1 (resp F_2) are applied in the two leftmost (resp. rightmost) figures.*

Let $\mathcal{M}_{rest}$ be a conformal tetrahedral mesh at its rest position. The initial position of each vertex is written as $\mathbf{p}_i$ while its position in the deformed position is written as $\mathbf{q}_i$ (see Figure 16). The displacement at each node is then defined as ;

$$\mathbf{u}_i = \mathbf{q}_i - \mathbf{p}_i$$

We use a linear tetrahedron finite element, denoted in the literature as P_1. This amounts to assuming a C^0 continuity of the displacement vector across the domain and equivalently assuming constant strain inside each tetrahedron (since the gradient matrix is constant inside each tetrahedron).
More precisely, if $\mathcal{T}$ is a tetrahedron defined by its four vertices $\mathbf{p}_j$, $j = 0, \ldots, 3$, in their **rest position**, then the displacement vector at a given point $\mathbf{X} = (x, y, z) \subset \mathcal{T}$ is defined as :

$$\mathbf{u}(\mathbf{X}) = \sum_{j=0}^{3} h_j(\mathbf{X})\mathbf{u}_j$$

where $h_j(\mathbf{X})$, $j = 0, \ldots, 3$ are the shape functions that correspond to the linear interpolation inside tetrahedron $\mathcal{T}$. These shape functions $h_j(\mathbf{X})$ correspond to

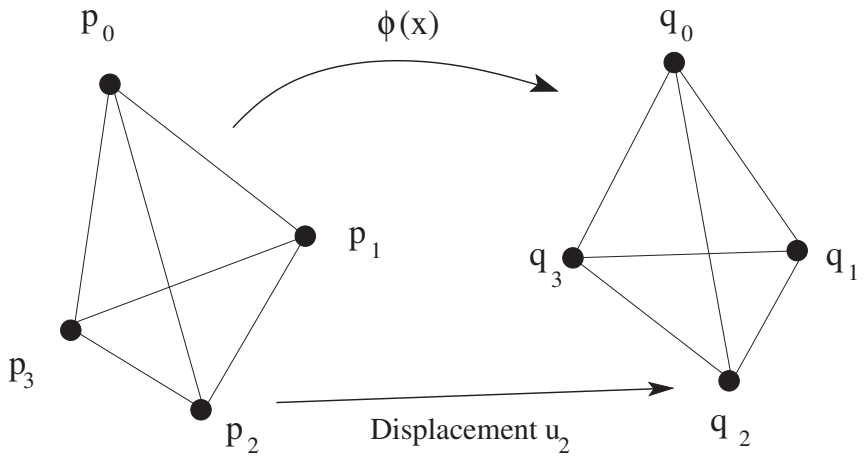


Figure 16: Notations for the position and displacement vectors of a tetrahedron.

the barycentric coordinates of point $\mathbf{X}$ with respect to vertices $\mathbf{p}_i$. The analytical expression of these shape functions is obtained from the linear relation :

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_0^x & \mathbf{p}_1^x & \mathbf{p}_2^x & \mathbf{p}_3^x \\ \mathbf{p}_0^y & \mathbf{p}_1^y & \mathbf{p}_2^y & \mathbf{p}_3^y \\ \mathbf{p}_0^z & \mathbf{p}_1^z & \mathbf{p}_2^z & \mathbf{p}_3^z \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \end{bmatrix} = \mathbf{P} \quad \mathbf{H}$$

where $\mathbf{p}_i = (\mathbf{p}_i^x, \mathbf{p}_i^y, \mathbf{p}_i^z)^T$ are the coordinates of each tetrahedron vertex. The matrix $\mathbf{P}$ completely encapsulates the shape of the tetrahedron $\mathcal{T}$ at its rest position. Since its determinant $|\mathbf{P}| = 6V(\mathcal{T})$ is the volume of $\mathcal{T}$, for non degenerate tetrahedra $\mathbf{P}$ can be inverted :

$$\left[\mathbf{P}^{-1}\right] = \frac{-1}{6V(\mathcal{T})} \begin{bmatrix} \mathbf{m}_0^x & \mathbf{m}_0^y & \mathbf{m}_0^z & -V_0 \\ \mathbf{m}_1^x & \mathbf{m}_1^y & \mathbf{m}_1^z & -V_1 \\ \mathbf{m}_2^x & \mathbf{m}_2^y & \mathbf{m}_2^z & -V_2 \\ \mathbf{m}_3^x & \mathbf{m}_3^y & \mathbf{m}_3^z & -V_3 \end{bmatrix}$$

where :

- $\mathbf{m}_i = (m_i^x, m_i^y, m_i^z)^T$ is the ith area vector opposite to vertex $\mathbf{p}_i$ (see description below),
- $V_i = (-1)^{i+1} \ |\mathbf{p}_{i+1}, \mathbf{p}_{i+2}, \mathbf{p}_{i+3}|$ can be interpreted[5] geometrically as 6 times the volume of the tetrahedron made by the origin $\mathbf{o}$ and vertices $\mathbf{p}_{i+1}$, $\mathbf{p}_{i+2}$ and $\mathbf{p}_{i+3}$. To simplify notations, the index $i + k$ should be understood as $(i + k) mod\ 4$.

Area vectors $\mathbf{m}_i$ have a very simple interpretation : they are directed along the outer normal direction of the triangle T_i opposite to $\mathbf{p}_i$ and their norm is equal

[5] $|\mathbf{a}, \mathbf{b}, \mathbf{c}|$ is the triple product of vectors $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$

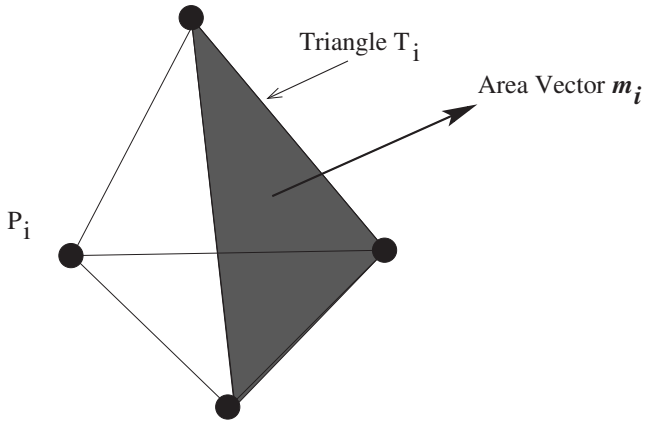


Figure 17: Definition of area vector $\mathbf{m}_i$ on the triangle T_i opposite to vertex $\mathbf{p}_i$ in tetrahedron $\mathcal{T}$

to twice the area of that triangle (see Figure 17). More precisely, they can be computed as :

$$\mathbf{m}_i = (-1)^{i+1} \left(\mathbf{p}_{i+1} \times \mathbf{p}_{i+2} + \mathbf{p}_{i+2} \times \mathbf{p}_{i+3} + \mathbf{p}_{i+3} \times \mathbf{p}_{i+1} \right) \tag{11}$$

where $\mathbf{p}_{i+1} \times \mathbf{p}_{i+2}$ is the cross product between the two vectors $\mathbf{p}_{i+1}$ and $\mathbf{p}_{i+2}$. Because they are computed from the inverse of matrix $\mathbf{P}$, these area vectors also capture the shape of $\mathcal{T}$ completely, and thus play a key role when computing the stiffness matrix $\mathbf{K}$. Further properties of area vectors are described in section 4.2.
The shape functions $h_i(\mathbf{X})$ can then be written as :

$$h_i(\mathbf{X}) = -\frac{\mathbf{m}_i \cdot \mathbf{X} - V_i}{6V(\mathcal{T})} \tag{12}$$

where $\mathbf{m}_i \cdot \mathbf{X}$ is the dot product between the two vectors $\mathbf{m}_i$ and $\mathbf{X}$.
If we note that elementary volumes V_i can be expressed as :

$$V_i = \mathbf{m}_i \cdot \mathbf{p}_{i+1}$$

then the interpolation of displacement vectors can be written as :

$$\mathbf{u}(\mathbf{X}) = -\sum_{i=0}^{3} \frac{\mathbf{m}_i \cdot (\mathbf{X} - \mathbf{p}_{i+1})}{6V(\mathcal{T})} \, \mathbf{u}_i \tag{13}$$

Finally, the interpolation matrix $\mathbf{H}(\mathbf{X})$ widely used in the finite element literature is defined as :

$$\mathbf{u}(\mathbf{X}) = \mathbf{H}(\mathbf{X}) \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \end{bmatrix}$$

$$\mathbf{H}(\mathbf{X}) = \begin{bmatrix} h_0 & 0 & 0 & h_1 & 0 & 0 & h_2 & 0 & 0 & h_3 & 0 & 0 \\ 0 & h_0 & 0 & 0 & h_1 & 0 & 0 & h_2 & 0 & 0 & h_3 & 0 \\ 0 & 0 & h_0 & 0 & 0 & h_1 & 0 & 0 & h_2 & 0 & 0 & h_3 \end{bmatrix}$$

4.2 Properties of area vectors

Area vectors have a major significance with respect to the geometry of a tetrahedron for instance through the law of cosine. To write essential geometric relations, we need to introduce the following quantities :

- *normal vector* $\mathbf{n}_i$ of triangle T_i defined as the normalized area vector : $\mathbf{n}_i = \frac{\mathbf{m}_i}{\|\mathbf{m}_i\|}$. The normal vector is pointing outward if the tetrahedron $\mathcal{T}$ is positively oriented, i.e. if its volume $V(\mathcal{T})$ is positive.

- *dihedral angle* $\theta_{i,j}$ existing between triangle T_i and T_j and therefore between their normal vectors $\mathbf{n}_i$ and $\mathbf{n}_j$.

- *triangle area* A_i, area of triangle T_i

- *edge length* $l_{i,j}$ is the length between vertex $\mathbf{p}_i$ and $\mathbf{p}_j$ (see Figure 18).

- *foot height* f_i is the height of vertex $\mathbf{p}_i$ above triangle T_i (see Figure 18).

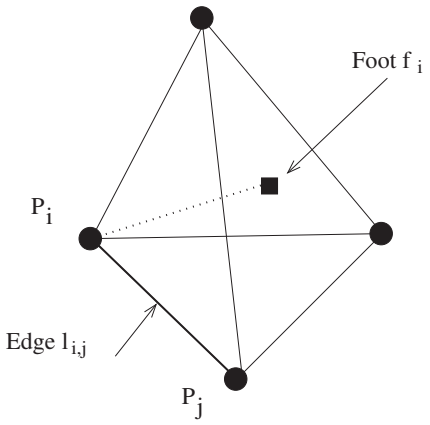


Figure 18: Definition of foot height f_i and edge length $l_{i,j}$ in tetrahedron $\mathcal{T}$

The definition of area vectors gives the relation :

$$\mathbf{m}_i = 2A_i\mathbf{n}_i$$

Noting that the tetrahedron volume is simply related to the foot height and area, we get :

$$\mathbf{m}_i = \frac{2V(\mathcal{T})}{3f_i}\mathbf{n}_i$$

From the relations $\left[\mathbf{P}^{-1}\right]\left[\mathbf{P}\right] = \mathbf{I}_3$ and $\left[\mathbf{P}\right]\left[\mathbf{P}^{-1}\right] = \mathbf{I}_3$, the following relations are obtained :

$$\sum_i \mathbf{m}_i = 0 \tag{14}$$

$$\sum_i \mathbf{m}_i \cdot \mathbf{p}_i = -18V(\mathcal{T}) \tag{15}$$

$$\sum_i \mathbf{m}_i \cdot \mathbf{p}_{i+1} = 6V(\mathcal{T})$$

$$(\mathbf{p}_{i+1} - \mathbf{p}_i) \cdot \mathbf{m}_i = 6V(\mathcal{T})$$

$$\sum_{i\neq j, i<j} \mathbf{m}_i \cdot \mathbf{m}_j l_{i,j}^2 = 108V(\mathcal{T})^2$$

$$|\mathbf{m}_i, \mathbf{m}_{i+1}, \mathbf{m}_{i+2}| = (-1)^{i+1}\ 36V(\mathcal{T})^2$$

The most important result is that all area vectors sum to zero. A result of this property is the *law of cosine* :

$$A_0^2 = A_1^2 + A_2^2 + A_3^2 - 2A_1A_2cos\theta_{1,2} - 2A_1A_3cos\theta_{1,3} - 2A_2A_3cos\theta_{2,3} \tag{16}$$

In fact, area vectors sum to zero for any closed trianguled surface. Indeed, through Green's formulae [14], the sum of area vectors can be interpreted as the total flow of a constant field accross a closed surface.

4.3 Computation of Stiffness Matrix : Isotropic case

We use a displacement based finite element method which is equivalent to the classical Ritz analysis [6]. On a single tetrahedron, the (linear) isotropic elastic energy is equal to :

$$W(\mathcal{T}) = \int_{\mathcal{T}} \left(\frac{\lambda}{2}(tr\mathbf{E}_{\mathrm{L}})^2 + \mu\ tr\mathbf{E}_{\mathrm{L}}^2 \right) dV$$

The gradient of the displacement $\nabla\mathbf{u}(\mathbf{X})$ is constant inside $\mathcal{T}$:

$$\begin{aligned} \nabla\mathbf{u}(\mathbf{X}) &= -\sum_{i=0}^{3} \nabla \frac{\mathbf{m}_i \cdot (\mathbf{X} - \mathbf{p}_{i+1})}{6V(\mathcal{T})}\ \mathbf{u}_i \\ &= -\sum_{i=0}^{3} \frac{1}{6V(\mathcal{T})} \mathbf{m}_i \otimes \mathbf{u}_i \end{aligned}$$

where $\mathbf{m}_i \otimes \mathbf{u}_i = \mathbf{m}_i\mathbf{u}_i^T$ is the tensor product of the two vectors :

$$\mathbf{m}_i \otimes \mathbf{u}_i = \begin{bmatrix} \mathbf{m}_i^x\mathbf{u}_i^x & \mathbf{m}_i^x\mathbf{u}_i^y & \mathbf{m}_i^x\mathbf{u}_i^z \\ \mathbf{m}_i^y\mathbf{u}_i^x & \mathbf{m}_i^y\mathbf{u}_i^y & \mathbf{m}_i^y\mathbf{u}_i^z \\ \mathbf{m}_i^z\mathbf{u}_i^x & \mathbf{m}_i^z\mathbf{u}_i^y & \mathbf{m}_i^z\mathbf{u}_i^z \end{bmatrix}$$

The infinitesimal strain tensor $\mathbf{E}_{\mathrm{L}}$ is also constant inside $\mathcal{T}$:

$$\mathbf{E}_{\mathrm{L}} = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T) = \frac{-1}{12V(\mathcal{T})} \sum_{i=0}^{3} (\mathbf{m}_i \otimes \mathbf{u}_i + \mathbf{u}_i \otimes \mathbf{m}_i)$$

The first invariant $(tr\mathbf{E}_{\mathrm{L}})^2$ is simply :

$$(tr\mathbf{E}_{\mathrm{L}})^2 = \frac{1}{144V(\mathcal{T})^2}\left(\sum_i^3 \mathbf{m}_i \cdot \mathbf{u}_i\right)^2 = \frac{1}{144V(\mathcal{T})^2}\sum_{i,j=0}^{3} \mathbf{u}_i^T[\mathbf{m}_i \otimes \mathbf{m}_j]\mathbf{u}_j$$

The second invariant is slightly more complex to obtain :

$$\begin{aligned} tr\mathbf{E}_{\mathrm{L}}^2 &= \frac{1}{144V(\mathcal{T})^2} tr\Big(\sum_{i,j=0}^{3} (\mathbf{m}_i \otimes \mathbf{u}_j)(\mathbf{u}_i \cdot \mathbf{m}_j) + (\mathbf{u}_i \otimes \mathbf{m}_j)(\mathbf{m}_i \cdot \mathbf{u}_j) + \\ &\quad (\mathbf{m}_i \otimes \mathbf{m}_j)(\mathbf{u}_i \cdot \mathbf{u}_j) + (\mathbf{u}_i \otimes \mathbf{u}_j)(\mathbf{m}_i \cdot \mathbf{m}_j)\Big) \\ &= \frac{1}{72V(\mathcal{T})^2}\sum_{i,j=0}^{3} \mathbf{u}_i^T\left[(\mathbf{m}_j \otimes \mathbf{m}_i) + (\mathbf{m}_j \cdot \mathbf{m}_i)\mathbf{I}_3\right]\mathbf{u}_j \end{aligned}$$

Finally the linear elastic energy , is a quadratic function of the displacement and is written as :

$$\begin{aligned} W(\mathcal{T}) &= \frac{1}{72V(\mathcal{T})}\sum_{i,j=0}^{3} \mathbf{u}_i^T\left[\lambda(\mathbf{m}_i \otimes \mathbf{m}_j) + \mu(\mathbf{m}_j \otimes \mathbf{m}_i) + \mu(\mathbf{m}_i \cdot \mathbf{m}_j)\mathbf{I}_3\right]\mathbf{u}_j \\ W(\mathcal{T}) &= \frac{1}{2}\sum_{i,j=0}^{3} \mathbf{u}_i^T[\mathbf{B}_{ij}^{\mathcal{T}}]\mathbf{u}_j \end{aligned} \tag{17}$$

where $[\mathbf{B}_{ij}^{\mathcal{T}}]$ is the 3×3 stiffness matrix of tetrahedron $\mathcal{T}$ between vertices i and j. Noting that $[\mathbf{m}_i \otimes \mathbf{m}_j]\mathbf{a} = \mathbf{m}_i(\mathbf{m}_j \cdot \mathbf{a})$, we can write the local elastic energy in terms of dot products :

$$\begin{aligned} W(\mathcal{T}) &= \frac{1}{72V(\mathcal{T})}\sum_{i,j=0}^{3} \Big(\lambda(\mathbf{u}_i \cdot \mathbf{m}_i)(\mathbf{m}_j \cdot \mathbf{u}_j) + \mu(\mathbf{u}_i \cdot \mathbf{m}_j)(\mathbf{m}_i \cdot \mathbf{u}_j) + \\ &\quad \mu(\mathbf{m}_i \cdot \mathbf{m}_j)(\mathbf{u}_i \cdot \mathbf{u}_j)\Big) \end{aligned}$$

Since $\mathbf{m}_i \otimes \mathbf{m}_j = (\mathbf{m}_j \otimes \mathbf{m}_i)^T$, it is clear that local tensors are symmetric matrices : $[\mathbf{B}_{ij}^{\mathcal{T}}] = [\mathbf{B}_{ji}^{\mathcal{T}}]^T$. Therefore, there are only 10 distinct local stiffness matrices with *four vertex matrices* $[\mathbf{B}_{ii}^{\mathcal{T}}]$ and six *edge matrices* $[\mathbf{B}_{ij}^{\mathcal{T}}]$, $i \neq j$.

4.3.1 Local Vertex Stiffness Matrix

Vertex stiffness matrices take the simple form with normal vector $\mathbf{n}_i$:

$$[\mathbf{B}_{ii}^{\mathcal{T}}] = \frac{A_i^2}{9V(\mathcal{T})}\left((\lambda+\mu)(\mathbf{n}_i \otimes \mathbf{n}_i) + \mu\mathbf{I}_3\right) \tag{18}$$

These matrices have eigenvalues $(\lambda+2\mu, \mu, \mu)$, $\mathbf{n}_i$ being the first eigenvector, and two directions orthogonal to $\mathbf{n}_i$ being the two other eigenvectors.

4.3.2 Local Edge Stiffness Matrix

The stiffness matrix between vertex i and j is :

$$[\mathbf{B}_{ij}^{\mathcal{T}}] = \frac{1}{36V(\mathcal{T})}\left(\lambda(\mathbf{m}_i \otimes \mathbf{m}_j) + \mu(\mathbf{m}_j \otimes \mathbf{m}_i) + \mu(\mathbf{m}_i \cdot \mathbf{m}_j)\mathbf{I}_3\right) \tag{19}$$

This matrix has the edge direction $\frac{\mathbf{p}_j-\mathbf{p}_i}{\|\mathbf{p}_j-\mathbf{p}_i\|}$ as first eigenvector associated with eigenvalue $\frac{\mu(\mathbf{m}_i\cdot\mathbf{m}_j)}{36V(\mathcal{T})}$. The existence of the other two eigenvectors depends on the sign of the following matrix determinant :

$$\begin{vmatrix} (\lambda+\mu)(\mathbf{m}_i\cdot\mathbf{m}_j) & \mu\|\mathbf{m}_i\|^2 \\ \lambda\|\mathbf{m}_j\|^2 & 2\mu(\mathbf{m}_i\cdot\mathbf{m}_j) \end{vmatrix} = \lambda\mu A_i^2 A_v^2 \left(2(1+\frac{\lambda}{\mu})\cos^2\theta_{i,j} - 1\right)$$

4.3.3 Global Stiffness Matrix

The elastic energy of the whole deformed body is then computed by summing equation (17) over all tetrahedra. This total energy $W(\mathcal{M}_{def})$ may be written with the displacement vector $\mathbf{U}$, gathering all displacement vectors $\mathbf{u}_i$, and a global stiffness matrix $\mathbf{K}$:

$$W(\mathcal{M}_{def}) = \frac{1}{2}\mathbf{U}^T\mathbf{K}\mathbf{U} \tag{20}$$

This stiffness matrix $\mathbf{K}$ is built by assembling local stiffness matrices $[\mathbf{B}_{ij}^{\mathcal{T}}]$. Because these local matrices are symmetric with respect to the swap of indices $[\mathbf{B}_{ij}^{\mathcal{T}}] = [\mathbf{B}_{ji}^{\mathcal{T}}]$, the global stiffness matrix $\mathbf{K}$ is symmetric.

4.3.4 Global Vertex Stiffness Matrix

The 3×3 submatrix $[\mathbf{K}_{i,j}]$ associated with vertices i (row index) and j (column index) is computed as the sum of local stiffness matrices for all tetrahedra containing both vertices. The set of tetrahedra adjacent to a given vertex (*resp.* edge) is called the *shell* $\mathcal{S}(i)$ of this vertex (*resp.* edge). In particular for diagonal submatrices, we get :

$$[\mathbf{K}_{i,i}] = \sum_{\mathcal{T}\in\mathcal{S}(i)} \frac{1}{36V(\mathcal{T})}\left((\lambda_{\mathcal{T}}+\mu_{\mathcal{T}})(\mathbf{m}_i\otimes\mathbf{m}_i) + \mu_{\mathcal{T}}A_i^2\mathbf{I}_3\right)$$

In fact, we can provide a rather simple interpretation of this matrix expression. Its first term can be seen as the inertial matrix (second order moment) of area vectors $\mathbf{m}_i$ weighted by $\frac{(\lambda_{\mathcal{T}}+\mu_{\mathcal{T}})}{72V(\mathcal{T})}$ (see figure 19). Indeed, on a manifold tetrahedral mesh (but not on all conformal tetrahedral meshes), the shell of an interior vertex is homeomorphic to a sphere and it can be easily proved that the sum of its area vectors is null (Minkowsky's sum). Therefore $\mathbf{m}_i\otimes\mathbf{m}_i$ represents the local contribution to an inertia matrix. If vertex $\mathbf{p}_i$ is surrounded by semi-regular tetrahedra, then the matrix of inertia is proportional to identity. Note also that because it is weighted by the inverse of the tetrahedron's volume, it is very sensitive to the disparity in tetrahedra size. The second part is simply the sum of the second Lamé coefficient weighted with the inverse of the tetrahedron volume.

4.3.5 Global Edge Stiffness Matrix

The off-diagonal terms $[\mathbf{K}_{i,j}]$ of the stiffness matrix correspond to edge stiffness matrices that are the sum of local edge stiffness matrices. The edge direction $\frac{\mathbf{p}_j-\mathbf{p}_i}{\|\mathbf{p}_j-\mathbf{p}_i\|}$ is an eigenvector of $[\mathbf{K}_{i,j}]$ associated with the eigenvalue $k_{i,j}$:

$$k_{i,j} = \sum_{\mathcal{T}\in\mathcal{S}(i,j)} \frac{\mu_{\mathcal{T}}(\mathbf{m}_i\cdot\mathbf{m}_j)}{36V(\mathcal{T})}$$

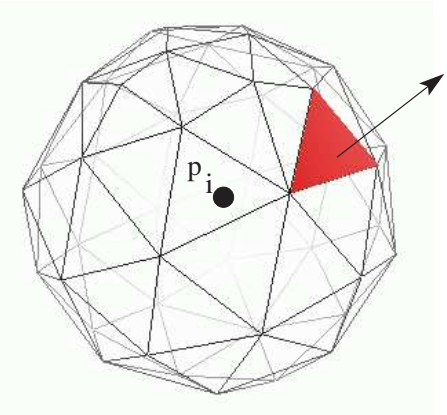


Figure 19: Shell of a vertex $\mathbf{p}_i$ in a tetrahedral mesh : only the opposite triangle is drawn. For each triangle the area vector $\mathbf{m}_i$ is pointing outward. The sum of these area vectors is null and their weighted matrix of inertia determines the global stiffness matrix at $\mathbf{p}_i$.

where $\mathcal{S}(i,j)$ is the set of tetrahedra adjacent to that edge (its shell). The tetrahedron volume $V(\mathcal{T})$ can be written as a function of triangle areas $V(\mathcal{T}) = \frac{2}{3l_{i,j}^{opp}} A_i A_j \sin\theta_{i,j}$ where $l_{i,j}^{opp}$ is the length of the opposite edge in tetrahedron $\mathcal{T}$ (see Figure 20).

$$k_{i,j} = \frac{1}{6} \sum_{\mathcal{T} \in \mathcal{S}(i,j)} \mu_{\mathcal{T}} \; l_{i,j}^{opp} \; \cot\theta_{i,j}$$

4.4 Physical Interpretation of Isotropic Stiffness Matrix

Equation (17) describes the local stiffness matrices and it can be interpreted as the sum of discrete differential operators. Indeed, the isotropic elastic energy related to the first invariant of the infinitesimal strain tensors, can be written as a quadratic functional of the displacement vector :

$$W(\mathbf{X}) = \frac{\lambda}{2} \; (tr\mathbf{E}_{\mathrm{L}})^2 + \mu \; tr\mathbf{E}_{\mathrm{L}}^2 = \frac{\lambda}{2}(\mathrm{d}iv \; \mathbf{u})^2 + \mu \; tr(\nabla\mathbf{u}^T \nabla\mathbf{u}) - \frac{\mu}{2}\| \, \mathbf{curl} \, \mathbf{u}\|^2$$

The first variation of the elastic force $-\delta W$ can be interpreted as the density of linear elastic force per unit volume, and is given by the Lamé equation :

$$-\delta W = (\lambda + \mu)\nabla(\mathrm{d}iv \; \mathbf{u}) + \mu\Delta\mathbf{u}$$

It is therefore natural to compare the Lamé equation with the expression of the discrete elastic force $\mathbf{F}_i(\mathcal{T})$ acting on a vertex i of tetrahedron $\mathcal{T}$:

$$\mathbf{F}_i(\mathcal{T}) = -\sum_{j=0}^{3} [\mathbf{B}_{ij}^{\mathcal{T}}]\mathbf{u}_j = \frac{-1}{36V(\mathcal{T})} \sum_{j=0}^{3} [\lambda(\mathbf{m}_i \otimes \mathbf{m}_j) + \mu(\mathbf{m}_j \otimes \mathbf{m}_i) + \mu(\mathbf{m}_i \cdot \mathbf{m}_j)\mathbf{I}_3] \, \mathbf{u}_j$$

The three terms of the local rigidity matrix may be interpreted as follows :

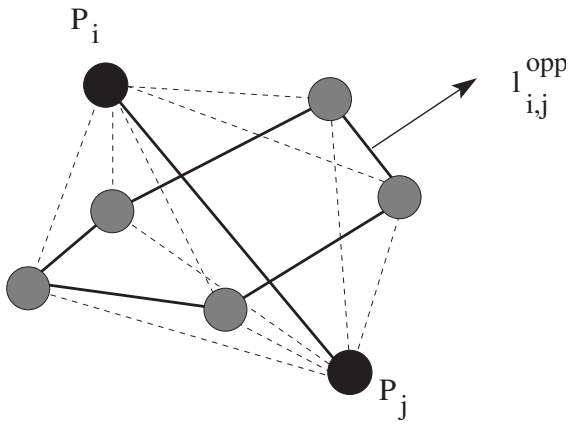


Figure 20: Shell of an edge linking vertices $\mathbf{p}_i$ and $\mathbf{p}_j$ in a tetrahedral mesh : the adjacent tetrahedra are drawn with dashed lines whereas opposite edges to that edge are drawn with solid lines. One of the eigenvalues of edge stiffness matrix depends on the weighted sum of the lengths $l_{i,j}^{opp}$.

$$\mathbf{F}_i(\mathcal{T}) = \frac{-1}{36V(\mathcal{T})} \sum_{j=0}^{3} \Big[\underbrace{\lambda(\mathbf{m}_i \otimes \mathbf{m}_j)}_{\substack{\mathrm{T}_1 \\ \nabla(\mathrm{d}iv\ \mathbf{u}) \\ operator}} + \underbrace{\mu(\mathbf{m}_j \otimes \mathbf{m}_i)}_{\substack{\mathrm{T}_2 \\ \nabla(\mathrm{d}iv\ \mathbf{u}) \\ \mathrm{p}seudo-operator}} + \underbrace{\mu(\mathbf{m}_i \cdot \mathbf{m}_j)\mathbf{I}_3}_{\substack{\mathrm{T}_3 \\ \Delta\mathbf{u} \\ operator}} \Big] \mathbf{u}_j$$

The first term of the local rigidity matrix, corresponds to the integral of the operator $\nabla(\mathrm{d}iv\ \mathbf{u})$ over a subdomain of tetrahedron $\mathcal{T}$. Indeed through Green's second formulae [14], the integral over a domain D of that operator can be evaluated along its boundary ∂D :

$$\int_D \nabla(\mathrm{d}iv\ \mathbf{u}) = \int_{\partial D} (\mathrm{d}iv\ \mathbf{u})\mathbf{n}\ dS$$

However the divergence operator is actually constant over $\mathcal{T}$ and is equal to :

$$\frac{-1}{6V(\mathcal{T})} \sum_{j=0}^{3} \mathbf{m}_j \cdot \mathbf{u}_j$$

Furthermore, since the integral $\int \mathbf{n}\ dS$ over each triangle of $\mathcal{T}$ is equal to $\frac{1}{2}\mathbf{m}_i$, T_1 is equal to one third of the flux of $(\mathrm{d}iv\mathbf{u}\ \mathbf{n})$ through the face of $\mathcal{T}$ opposite to vertex $\mathbf{p}_i$:

$$T_1 = \frac{-1}{36V(\mathcal{T})} \sum_{j=0}^{3} [(\mathbf{m}_i \otimes \mathbf{m}_j)]\mathbf{u}_j = \frac{1}{3}\ \frac{1}{6V(\mathcal{T})} \left(\sum_{j=0}^{3} \mathbf{m}_j \cdot \mathbf{u}_j \right) \frac{\mathbf{m}_i}{2}$$

Thus, we can provide a straightforward interpretation on the first term of the local rigidity tensor : it corresponds to the integration of the $\nabla(\mathrm{d}iv\ \mathbf{u})$ operator

over a subdomain $\mathcal{D}_i$ for which $\int_{\mathcal{T}\ \mathcal{D}_i} \mathbf{n}\ dS = \frac{\mathbf{m}_i}{6}$. A natural choice for this subdomain is to consider the shell $\mathcal{S}_i$ of vertex $\mathbf{p}_i$ homothetically scaled down by a ratio of $1/\sqrt{3}$. This subdomain is sketched in Figure 21 (a) and (b): the vertices of the subdomain are located at distance of $1/\sqrt{3}$ times the original edge length from vertex $\mathbf{p}_i$. Unfortunately since $1/\sqrt{3} > 0.5$, the subdomain of two neighboring vertices overlap.

To obtain a non-overlapping subdomain $\mathcal{D}_i$, one should consider the subdomain defined by the middle of each edge, the barycenters of each triangles, and the barycenter of the tetrahedron, as proposed by Putti *et al.* [88]. More precisely, as shown in Figure 21 (c), the subdomain consists in the six triangles $(FAG, GAB, BGC, CGD, DGE, EGF)$ where A, C, E are the centers of the three triangles adjacent to $\mathbf{p}_i$, B, D, F are the centers of the three adjacent edges and G is the tetrahedron barycenter. This subdomain is called *the barycentric dual cell* in [19].

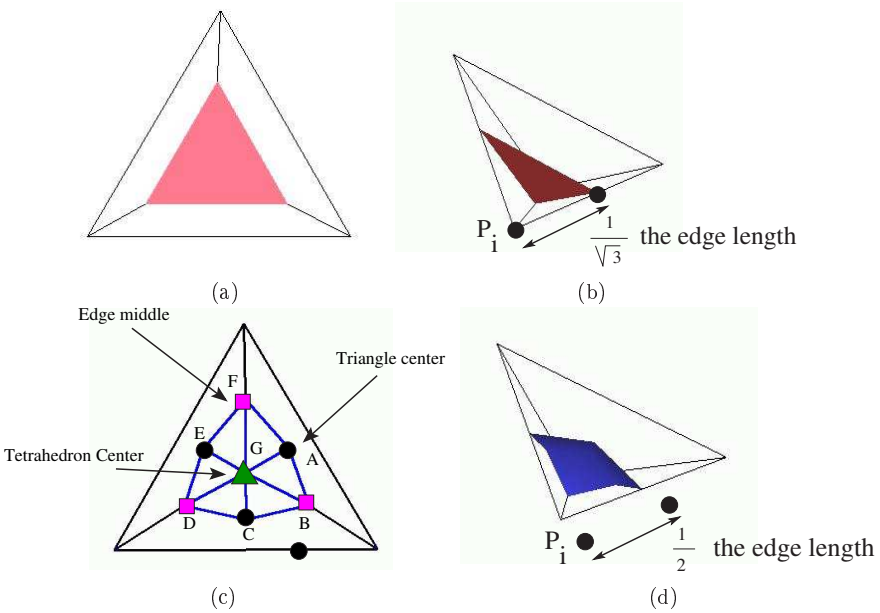


Figure 21: Definitions of two subdomains for which $\int \mathbf{n}\ dS$ is equal to one third the value through triangle $\triangle_i$, opposite of vertex $\mathbf{p}_i$ in tetrahedron $\mathcal{T}$; (a) and (b) : front and side view of the first subdomain consisting of a single triangle corresponding to the homothety $\triangle_i$ with a ratio of $\frac{1}{\sqrt{3}}$; (c) and (d) front and side view of the second non-overlapping subdomain consisting of 6 triangles linking the edge middles, triangle centers and tetrahedron center.

Indeed, the flux over the six triangles may be written as a sum of cross products :

$$\int_{\mathcal{D}_i} \mathbf{n}\ dS = A \times B + B \times C + C \times D + D \times E + E \times F$$

Since A, B, C, D, E, F, G are simple barycentric coordinates of the four tetrahedron vertices $\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k, \mathbf{p}_l$, it can be simply evaluated as a function of these

vertices :

$$\int_{\mathcal{D}_i} \mathbf{n}\ dS = \frac{1}{6}\left(\mathbf{p}_j \times \mathbf{p}_k + \mathbf{p}_k \times \mathbf{p}_l + \mathbf{p}_l \times \mathbf{p}_j\right) = \frac{\mathbf{m}_i}{6}$$

Thus, to summarize, we have proved so far that term T_1 is the integral of the $\nabla(div\ \mathbf{u})$ operator over a non-overlapping subdomain centered on $\mathbf{p}_i$.
The second term T_2 of the local rigidity matrix is the transposed matrix of the first term T_1 but cannot be interpreted in terms of a linear differential operator. In fact, if we write T_2 as $\sum(\mathbf{m}_i \cdot \mathbf{u}_j)\mathbf{m}_j$ we can state that T_2 corresponds to the flux of a scalar field equal to $\frac{1}{12V(\mathcal{T})}(\mathbf{m}_i \cdot \mathbf{u}_j)$ over each face of the subdomain $\mathcal{D}_i$. It should be noticed that T_2 has no equivalent in the continuous formulation (the Lamé equation) and is produced by the evaluation of $\|\,\mathbf{curl}\,\mathbf{u}\|^2$.
The third term T_3 corresponds to the discrete Laplacian operator and its expression originates from the evaluation of $\frac{1}{2}tr(\nabla\mathbf{u}^T\nabla\mathbf{u})$. The same approach as for the $\nabla(div\ \mathbf{u})$ can be applied. First, the integral of the Laplacian operator is integrated over a domain D using the integral Gauss theorem. For the x component u^x of the displacement field, it gives :

$$\int_D \Delta\ u^x\ dV = \int_D \nabla \cdot (\nabla\ u^x)\ dV = \int_{\partial D} (\nabla\ u^x) \cdot \mathbf{n}\ dS$$

If the domain D is included inside a tetrahedron, then the gradient of the displacement field is a constant vector :

$$\nabla u^x = \frac{-1}{6V(\mathcal{T})} \sum_{j=0}^{3} \mathbf{m}_j u_j^x$$

If we suppose that the domain boundary coincides with triangle $\triangle_i$, opposite to $\mathbf{p}_i$ in tetrahedron $\mathcal{T}$, then we get :

$$\int_D \Delta\ u^x\ dV = \frac{-1}{6V(\mathcal{T})} \sum_{j=0}^{3} \mathbf{m}_j \frac{1}{6V(\mathcal{T})} \sum_{j=0}^{3} \mathbf{m}_j u_j^x \cdot \frac{\mathbf{m}_i}{2}$$

Therefore, T_3 corresponds to the integral of the Laplacian operator over a domain $\mathcal{D}$ for which $\int_{\partial\mathcal{D}\mathcal{T}} \mathbf{n}\ dS = \frac{\mathbf{m}_i}{2}$, for instance the subdomain defined in Figure 21 (c), corresponding to the barycentric dual cell of vertex $\mathbf{p}_i$ in tetrahedron $\mathcal{T}$. The Finite Element approximation of the Laplacian operator on tetrahedra was previously studied by Putti *et al.* [88], Debunne [28] and Cosmi [19].
To summarize, we have proved that the *variational formulation* of linear elasticity over tetrahedral meshes is not completely equivalent to the Finite Difference and Finite Volume methods. Indeed, the latter methods are equivalent to the *differential formulation* of Finite Element method which leads to the following equation of the elastic force :

$$\mathbf{F}_i(\mathcal{T}) = \frac{-1}{36V(\mathcal{T})} \sum_{j=0}^{3} \left[(\lambda+\mu)(\mathbf{m}_i \otimes \mathbf{m}_j) + \mu(\mathbf{m}_i \cdot \mathbf{m}_j)\mathbf{I}_3\right] \mathbf{u}_j \qquad (21)$$

The variational formulation of the FEM creates the stiffness matrix from the expression of the elastic energy whereas the differential formulation of the FEM is based on the Lamé differential equation.

4.5 Computation of Stiffness Matrix : Transversally Anisotropic case

From section 3.2.4, the density of elastic energy for a transversally anisotropic material can be derived from the isotropic case by adding a corrective term :

$$\begin{aligned} W(\mathbf{X})_{\mathrm{T}ransv.Ani} &= W(\mathbf{X}) + \Delta W_{\mathrm{A}ni}(\mathbf{X}) \\ W(\mathbf{X})_{\mathrm{T}ransv.Ani} &= W(\mathbf{X}) + \Delta\lambda I_1 I_4 + 2\Delta\mu\ I_5 - \left(\frac{\Delta\lambda}{2} + \Delta\mu\right) I_4^2 \end{aligned}$$

where $\Delta\lambda$ and $\Delta\mu$ are the variation of Lamé coefficient in the direction of anisotropy $\mathbf{a}_0$ and where I_4 and I_5 are the constants defined in equations 7 and 8. The evaluation of I_4 and I_5 with the linear tetrahedron finite element gives :

$$\begin{aligned} I_4 &= \frac{-1}{6V(\mathcal{T})} \sum_{i=0}^{3} (\mathbf{a}_0 \cdot \mathbf{m}_i)(\mathbf{a}_0 \cdot \mathbf{u}_i) \\ (\mathrm{tr}\mathbf{E}_{\mathrm{L}})I_4 &= \frac{1}{72V(\mathcal{T})^2} \mathbf{u}_i^T \left[(\mathbf{a}_0 \cdot \mathbf{m}_j)(\mathbf{m}_i \otimes \mathbf{a}_0)\right] \mathbf{u}_j \\ I_4^2 &= \frac{1}{36V(\mathcal{T})^2} \sum_{i,j=0}^{3} \mathbf{u}_i^T \left[(\mathbf{a}_0 \cdot \mathbf{m}_i)(\mathbf{a}_0 \cdot \mathbf{m}_j)(\mathbf{a}_0 \otimes \mathbf{a}_0)\right] \mathbf{u}_j \end{aligned}$$

Similarly for I_5 :

$$\begin{aligned} I_5 = \frac{1}{144V(\mathcal{T})^2} \sum_{i,j=0}^{3} \mathbf{u}_i^T \quad & \Big[(\mathbf{a}_0 \cdot \mathbf{m}_j)(\mathbf{a}_0 \otimes \mathbf{m}_i) + (\mathbf{a}_0 \cdot \mathbf{m}_i)(\mathbf{m}_j \otimes \mathbf{a}_0) + \\ & (\mathbf{m}_i \cdot \mathbf{m}_j)(\mathbf{a}_0 \otimes \mathbf{a}_0) + (\mathbf{a}_0 \cdot \mathbf{m}_i)(\mathbf{a}_0 \cdot \mathbf{m}_j)\mathbf{I}_3\Big] \mathbf{u}_j \end{aligned}$$

Thus the additional elastic energy term $\Delta W(\mathcal{T})_{\mathrm{A}ni}$ due to transversal anisotropy can also be written as a bilinear function of vertex displacements :

$$\Delta W(\mathcal{T})_{\mathrm{A}ni} = \frac{1}{2} \sum_{i,j=0}^{3} \mathbf{u}_i^T [\mathbf{A}_{ij}^{\mathcal{T}}] \mathbf{u}_j \tag{22}$$

with the local 3×3 matrix $[\mathbf{A}_{ij}^{\mathcal{T}}]$ being defined as :

$$\begin{aligned} [\mathbf{A}_{ij}^{\mathcal{T}}] = \frac{1}{144V(\mathcal{T})} \Big(& \Delta\lambda(\mathbf{a}_0 \cdot \mathbf{m}_j)(\mathbf{m}_i \otimes \mathbf{a}_0) - (\Delta\lambda + 2\Delta\mu)(\mathbf{a}_0 \cdot \mathbf{m}_i)(\mathbf{a}_0 \cdot \mathbf{m}_j)(\mathbf{a}_0 \otimes \mathbf{a}_0) + \\ & \Delta\mu(\mathbf{a}_0 \cdot \mathbf{m}_j)(\mathbf{a}_0 \otimes \mathbf{m}_i) + \Delta\mu(\mathbf{a}_0 \cdot \mathbf{m}_i)(\mathbf{m}_j \otimes \mathbf{a}_0) + \Delta\mu(\mathbf{m}_i \cdot \mathbf{m}_j)(\mathbf{a}_0 \otimes \mathbf{a}_0) + \\ & \Delta\mu(\mathbf{a}_0 \cdot \mathbf{m}_i)(\mathbf{a}_0 \cdot \mathbf{m}_j)\mathbf{I}_3 \Big) \end{aligned}$$

4.5.1 Local Vertex Stiffness Matrix

When $i = j$, the vertex stiffness matrix is written as :

$$\begin{aligned} [\mathbf{A}_{ii}^{\mathcal{T}}] = \frac{1}{144V(\mathcal{T})} \Big[& (\Delta\lambda + \Delta\mu)(\mathbf{a}_0 \cdot \mathbf{m}_i)(\mathbf{m}_i \otimes \mathbf{a}_0) - (\Delta\lambda + 2\Delta\mu)(\mathbf{a}_0 \cdot \mathbf{m}_i)^2(\mathbf{a}_0 \otimes \mathbf{a}_0) + \\ & \Delta\mu(\mathbf{a}_0 \cdot \mathbf{m}_i)(\mathbf{a}_0 \otimes \mathbf{m}_i) + \Delta\mu\|\mathbf{m}_i\|^2(\mathbf{a}_0 \otimes \mathbf{a}_0) + \Delta\mu(\mathbf{a}_0 \cdot \mathbf{m}_i)^2\mathbf{I}_3 \Big] \end{aligned}$$

This matrix has $\mathbf{c}_0$, the unit vector orthogonal to both $\mathbf{a}_0$ and $\mathbf{m}_i$ as first eigenvector with eigenvalue $\Delta\mu(\mathbf{a}_0 \cdot \mathbf{m}_i)^2$. The existence of the other two eigenvectors, in the plane defined by $\mathbf{a}_0$ and $\mathbf{m}_i$, depends on the sign of $(2\Delta\mu + \Delta\lambda)(\mathbf{a}_0 \cdot \mathbf{m}_i)^2 - \Delta\lambda\|\mathbf{m}_i\|^2$.

4.5.2 Global Stiffness Matrix

For a transversally anisotropic material, the global stiffness matrix $\mathbf{K}$ is assembled as the sum of local isotropic and anisotropic stiffness matrices :

$$[\mathbf{K}_{i,j}] = \sum_{\mathcal{T} \in \mathcal{S}(i,j)} [\mathbf{B}_{i,j}] + [\mathbf{A}_{i,j}] \tag{23}$$

One should note that the global matrix $[\mathbf{K}_{i,j}]$ contains non-null values only if vertices i and j are linked by an edge of the tetrahedral mesh.

4.6 Work of gravity forces

We calculate the potential energy of gravity forces when a displacement field $\mathbf{u}(\mathbf{X})$ is applied on the body $\mathcal{M}_{def}$. If we write $\mathbf{g}$ the gravity vector ($\|\mathbf{g}\| = 9.8m/s^2$), and ρ the density of the material (assumed constant for the whole body), then the potential energy of a tetrahedron $\mathcal{T}$ is a simple function of the center of mass $\mathcal{T}$:

$$W_g(\mathcal{T}) = \int_{\mathcal{T}} \rho \mathbf{X} \cdot \mathbf{g} dV = \rho \int_{\mathcal{T}} \mathbf{X} dV \cdot \mathbf{g} = \rho V(\mathcal{T}) \frac{\mathbf{q}_0 + \mathbf{q}_1 + \mathbf{q}_2 + \mathbf{q}_3}{4} \cdot \mathbf{g}$$

If we drop the constant part of this energy, which is equivalent to consider the work of gravity forces when a displacement field $\mathbf{u}(\mathbf{X})$ is applied, then we get :

$$W_g(\mathcal{T}) = \rho V(\mathcal{T}) \frac{\mathbf{u}_0 + \mathbf{u}_1 + \mathbf{u}_2 + \mathbf{u}_3}{4} \cdot \mathbf{g} = \frac{\rho V(\mathcal{T}) \mathbf{g}}{4} \begin{bmatrix} \mathbf{u}_0 & \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

The potential energy of the whole model $\mathcal{M}_{def}$ is the dot product of the following two vectors :

$$W_g = \sum_{\mathcal{T}} W_g(\mathcal{T}) = \mathbf{U}^T \mathbf{R}^g = \mathbf{U}^T \begin{bmatrix} \cdots \\ \mathbf{r}_i^g \\ \cdots \end{bmatrix}$$

where $\mathbf{R}^g$ is a vector of size $3N$. More precisely, the sub-vector $\mathbf{r}_i^g$ of $\mathbf{R}^g$ associated with vertex i is proportional to the gravity vector, the coefficient being the volume of its neighboring tetrahedra :

$$\mathbf{r}_i^g = \rho \left(\sum_{\mathcal{T} \in \mathcal{S}(i)} \frac{V(\mathcal{T})}{4} \right) \mathbf{g} \tag{24}$$

4.7 Work of External Surface Pressure

Among external forces acting on deformable soft tissue models, we include a pressure force $\mathbf{f}_p$ which is applied on a part of its surface. We consider that such pressure force has a constant intensity $\|\mathbf{f}_p\| = p$ but its direction may be either constant (contact with a stream of gas) or directed along the surface normal (contact with a solid, fluid or gas at low speed). In the latter case, the force applied on a triangle T is :

$$\mathbf{f}_p(T) = p\ \mathbf{n}(T)$$

For a tetrahedral mesh, we consider that such constant pressure $\mathbf{f}_p$ is applied on a set $\mathcal{C}$ of surface triangles. If we consider a triangle $T \in \mathcal{C}$ consisting of vertices $(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k)$, the work of $\mathbf{f}_p$ on this triangle is :

$$W_p(T) = \int_T \mathbf{f}_p \cdot \mathbf{u}(\mathbf{X})\ dA = A(T)\ \mathbf{f}_p \cdot \left(\frac{\mathbf{u}_i + \mathbf{u}_j + \mathbf{u}_k}{3} \right)$$

The work of external surface pressure on the whole model $\mathcal{M}_{def}$ is then :

$$W_g = \mathbf{U}^T \mathbf{R}^p = \mathbf{U}^T \begin{bmatrix} \cdots \\ \mathbf{r}_i^p \\ \cdots \end{bmatrix} \tag{25}$$

where $\mathbf{r}_i^p$ is null if vertex $\mathbf{p}_i$ is not adjacent to any triangles in $\mathcal{C}$ and is proportional the sum of triangles area otherwise :

$$\mathbf{r}_i^p = \sum_{\substack{\mathbf{p}_i \in T \\ T \in \mathcal{C}}} \frac{A(T)\mathbf{f}_p(T)}{3}$$

If the pressure force is applied along the surface normal, then vector $\mathbf{r}_i^p$ has an intuitive formulation. The nodal force, resulting from the pressure applied on neighboring triangle, is proportional to the area sum of surrounding triangles and is directed along the surface normal $\mathbf{n}_i$ at vertex $\mathbf{p}_i$ (see Figure 22) :

$$\mathbf{r}_i^p = \frac{p}{3} \left(\sum_{\substack{\mathbf{p}_i \in T \\ T \in \mathcal{C}}} A(T) \right) \mathbf{n}_i$$

where $\mathbf{n}_i$ is computed as the average of surrounding triangle normals $\mathbf{n}(T)$ weighted by their area :

$$\mathbf{n}_i = \frac{\sum_{T \in \mathcal{C}} \mathbf{n}(T) A(T)}{\sum_{T \in \mathcal{C}} A(T)}$$

4.8 Mass Matrix

The mass matrix is derived from the evaluation of the kinetic energy $\mathcal{E}(\mathcal{M}_{def})$ on the whole body $\mathcal{M}_{def}$. The density of kinetic energy $w(\mathbf{X}) = \rho\ (\dot{\mathbf{u}}(\mathbf{X}))^2$

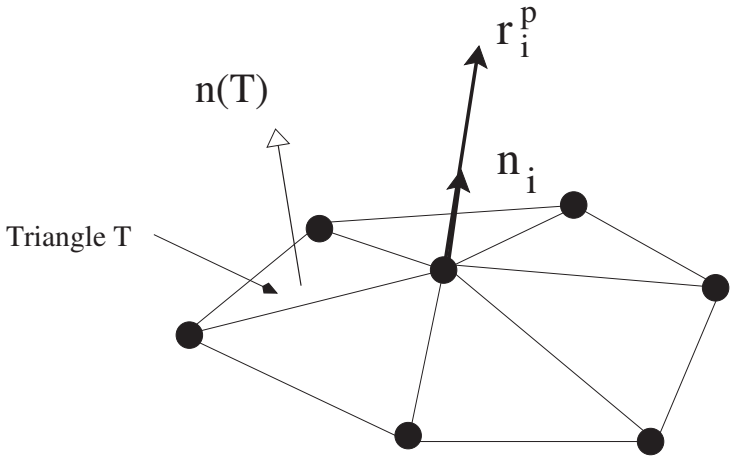


Figure 22: The pressure applied on neighboring triangles results in a force directed along the surface normal at a vertex and proportional to the sum of neighboring triangle area. The vertex surface normal $\mathbf{n}_i$ is computed as the weighted average of triangle normals.

where $\dot{\mathbf{u}}(\mathbf{X}) = \frac{d\mathbf{u}}{dt}$ is the speed of the material point $\mathbf{X}$. It follows that the kinetic energy of tetrahedron $\mathcal{T}$ is a bilinear form of the speed of nodal vertices $\dot{\mathbf{u}}_i$:

$$\mathcal{E}(\mathcal{T}) = \begin{bmatrix} \dot{\mathbf{U}}_0 \\ \dot{\mathbf{U}}_1 \\ \dot{\mathbf{U}}_2 \\ \dot{\mathbf{U}}_3 \end{bmatrix}^T \begin{bmatrix} \mathbf{M}^{\mathcal{T}}_{0,0} & \mathbf{M}^{\mathcal{T}}_{0,1} & \mathbf{M}^{\mathcal{T}}_{0,2} & \mathbf{M}^{\mathcal{T}}_{0,3} \\ \mathbf{M}^{\mathcal{T}}_{1,0} & \mathbf{M}^{\mathcal{T}}_{1,1} & \mathbf{M}^{\mathcal{T}}_{1,2} & \mathbf{M}^{\mathcal{T}}_{1,3} \\ \mathbf{M}^{\mathcal{T}}_{2,0} & \mathbf{M}^{\mathcal{T}}_{2,1} & \mathbf{M}^{\mathcal{T}}_{2,2} & \mathbf{M}^{\mathcal{T}}_{2,3} \\ \mathbf{M}^{\mathcal{T}}_{3,0} & \mathbf{M}^{\mathcal{T}}_{3,1} & \mathbf{M}^{\mathcal{T}}_{3,2} & \mathbf{M}^{\mathcal{T}}_{3,3} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{U}}_0 \\ \dot{\mathbf{U}}_1 \\ \dot{\mathbf{U}}_2 \\ \dot{\mathbf{U}}_3 \end{bmatrix}$$

This tetrahedron mass matrix has size 12×12 and is composed of 4×4 local mass matrix between vertex i and j, $\mathbf{M}^{\mathcal{T}}_{i,j}$ that are 3×3 diagonal matrices :

$$\mathbf{M}^{\mathcal{T}}_{i,j} = \rho \left(\int_{\mathcal{T}} h_i(\mathbf{X}) h_j(\mathbf{X}) dV \right) \mathbf{I}_3$$

To evaluate the integral, we use the 3 barycentric coordinates (h_0, h_1, h_2) as integration variables. Based on equations 13 and 14, the determinant of the Jacobian matrix is equal to the inverse of $6V(\mathcal{T})$:

$$\left| \begin{array}{ccc} \frac{\partial h_0}{\partial \mathbf{X}} & \frac{\partial h_1}{\partial \mathbf{X}} & \frac{\partial h_2}{\partial \mathbf{X}} \end{array} \right| = \frac{1}{216 V(\mathcal{T})^3} \left| \begin{array}{ccc} \mathbf{m}_0 & \mathbf{m}_1 & \mathbf{m}_2 \end{array} \right| = \frac{1}{6V(\mathcal{T})}$$

Thus the integral can be computed explicitely using the expression below :

$$\begin{aligned} \int_{\mathcal{T}} h_i(\mathbf{X}) h_j(\mathbf{X}) \, dV &= 6V(\mathcal{T}) \int_0^1 \int_0^{1-h_0} \int_0^{1-h_0-h_1} h_i \, h_j dh_0 \, dh_1 \, dh_2 \\ &= \frac{V}{10} \quad \text{if i=j} \\ &= \frac{V}{20} \quad \text{if } i \neq j \end{aligned}$$

Thus the local mass matrix $\mathbf{M}_{i,j}^{\mathcal{T}}$ is equal to $\frac{\rho V(\mathcal{T})}{10}\mathbf{I}_3$ if $i = j$ and to $\frac{\rho V(\mathcal{T})}{20}\mathbf{I}_3$ otherwise. If we perform *mass lumping* by considering only diagonal elements equal to the sum of row values, then we naturally get $\frac{\rho V(\mathcal{T})}{4}\mathbf{I}_3$, as if the tetrahedron mass is evenly spread over its four vertices.
The kinetic energy of the whole body can be written as a function of the global mass matrix built by assembling the local matrices $\mathbf{M}_{i,j}^{\mathcal{T}}$:

$$\mathcal{E}(\mathcal{M}_{def}) = \frac{1}{2}\dot{\mathbf{U}}^T\mathbf{M}\dot{\mathbf{U}} = \frac{1}{2}\dot{\mathbf{U}}^T[\mathbf{M}_{i,j}]\dot{\mathbf{U}}$$

where $\mathbf{M}_{i,j}$, the global 3×3 mass matrix between vertex i and j, depends on the volumes of tetrahedra adjacent to vertex i (if $i = j$) or tetrahedra adjacent to edge (i, j) if $i \neq j$:

$$\mathbf{M}_{i,i} = \rho \sum_{\mathcal{T}\in\mathcal{S}(i)} \frac{V(\mathcal{T})}{10}\mathbf{I}_3 \tag{26}$$

$$\mathbf{M}_{i,j} = \rho \sum_{\mathcal{T}\in\mathcal{S}(i,j)} \frac{V(\mathcal{T})}{20}\mathbf{I}_3 \quad \text{if } i \neq j \tag{27}$$

If we perform *mass lumping* to get a diagonal mass matrix $\mathbf{M}$ (and therefore easily invertible), then the vertex mass is equal to one fourth of the mass of its adjacent tetrahedra :

$$(\mathbf{M}_{i,i})_{\text{lumping}} = \rho \sum_{\mathcal{T}\in\mathcal{S}(i)} \frac{V(\mathcal{T})}{4}\mathbf{I}_3$$

4.9 Boundary Conditions

In a surgical simulator, the boundary conditions of a soft tissue model are related to the existence of contacts with either its neighboring anatomical structures or with surgical tools.
We simplify the interaction with other physical material by considering that such an interaction can be represented either in terms of imposed displacements or elastic forces or surface pressure forces. If the material is stiff, or if it is significantly stiffer than the material of interest, we model the contact by imposing given displacements on a set of vertices. For instance, in the case of the liver model, we consider that vertices located near the veina cava (a stiff vessel) are stable (zero displacement).
If neighboring materials are as stiff (or less) than the material of interest, then we model the interaction as a linearized spring force. More precisely for a boundary vertex $\mathbf{p}_i$, the applied force $\mathbf{r}_i^e$ is directed along a given direction $\mathbf{d}$, with stiffness k_e and rest displacement $\mathbf{u}_i^e$:

$$\mathbf{r}_i^e = -k_e\left((\mathbf{u}_i - \mathbf{u}_i^e)\cdot\mathbf{d}\right)\mathbf{d} = -k_e(\mathbf{d}\otimes\mathbf{d})(\mathbf{u}_i - \mathbf{u}_i^e) \tag{28}$$

Using a linearized spring allows to compute the static equilibrium by solving a linear system of equation. Indeed, the stiffness caused by the spring $k_e(\mathbf{d}\otimes\mathbf{d})$ can be added to the global stiffness matrix while the residual force $k_e(\mathbf{d}\otimes\mathbf{d})\mathbf{u}_i^e$ is added to the nodal load at node i. Furthermore, since the stiffness k_e is lower

than the Young modulus of the material, the condition number of the updated stiffness matrix is not significantly modified.
In the sequel, we do not consider linearized spring boundary conditions explicitely. Instead, we modify the global stiffness matrix $\mathbf{K}$ into $\mathbf{K}^\star$, and we consider that a nodal force $\mathbf{r}_i^b$ is applied to vertex $\mathbf{p}_i$:

$$[\mathbf{K}^\star_{i,i}] = [\mathbf{K}_{i,i} + k_e(\mathbf{d} \otimes \mathbf{d})]$$
$$\mathbf{r}_i^b = k_e(\mathbf{d} \otimes \mathbf{d})\mathbf{u}_i^e$$

When a soft tissue model is in contact with some fluids (bile, water, blood,...) or gas (carbon dioxide, air, ...) we make the hypothesis that a constant pressure is applied along the normal direction of the contact surface. The computation of the nodal forces is detailed in section 4.7.
Finally, the contact between surgical tools and a soft tissue model may be posed, in theory, either as imposed displacements (geometric method [6]) or as prescribed forces (penalty method [6]). However, in practice, the motion of surgical tools is controlled by the end-user through a force-feedback device. To decrease their cost, these devices are force-controlled and follow a simple open loop : the positions of surgical tools can be sent to a computer while they receive the force level that should be felt by the end-user. In other words, despite the low speed of a surgeon hands the position of a surgical tool varies significantly between two iterations ($dt = 20ms$) and therefore we found that the penalty method was not suited for deforming a soft tissue model.
Thus, after detecting the collision between soft tissue models and surgical tools, a set of imposed displacements at the collision nodes is computed. This computation is obviously ill-posed since it relies only on geometry (surface-volume intersection) rather than physical principles (Coulomb friction for instance). Furthermore, a major challenge is to design a stable contact algorithm where a small displacement of the tool entails a small variation of node position. The geometric contact algorithm used in our hepatic surgery simulator, can be found in [84].
To summarize, we consider only 2 types of boundary conditions in the remainder :

1. **Imposed Displacement** We write $\mathcal{V}_d$ the set of vertices $\mathbf{p}_i$ for which the displacement $\mathbf{u}_i^b$ is known. In the scope of surgery simulation, these vertices are always lying on the surface of the mesh.

2. **Applied Nodal Forces** We write $\mathcal{V}_f$ the set of vertices $\mathbf{p}_i$ where an external force r_i^b is applied. Again, we make the hypothesis that applied forces may exist only on surface nodes.

4.10 Equilibrium equations

We apply the principle of virtual displacements described in section 3.2.5 to obtain the finite element formulation of equilibrium equations. In a first stage, we only consider the static equilibrium by neglecting inertial forces. Thus, based on equation 10, we can state that the virtual elastic energy is equal to the sum of the virtual work of gravity and boundary forces :

$$\frac{1}{2}\hat{\mathbf{U}}^T\mathbf{K}\hat{\mathbf{U}} = \hat{\mathbf{U}}^T\mathbf{R}^g + \hat{\mathbf{U}}^T\mathbf{R}^b$$

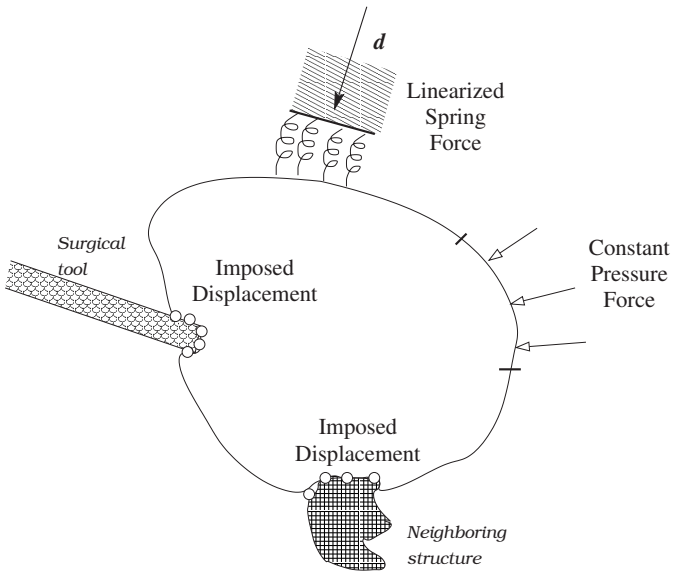


Figure 23: The three different boundary conditions resulting from interaction with neighboring structures or with surgical tools.

Since this equation must hold for any set of compatible displacements, the static equation of equilibrium becomes :

$$\mathbf{K}\mathbf{U} = \mathbf{R}^g + \mathbf{R}^b \tag{29}$$

It is important to note that equation 29 is written for all nodes including the $\mathcal{V}_d$ nodes where the displacement is imposed. Therefore, in order to compute the unknown displacement vectors (where no displacement is imposed), it is important to write equation 29 with a distinction between free nodes (subscript f) and constrained nodes (subscript c) :

$$\begin{bmatrix} \mathbf{K}_{ff} & \mathbf{K}_{fc} \\ \mathbf{K}_{cf} & \mathbf{K}_{cc} \end{bmatrix} \begin{bmatrix} \mathbf{U}_f \\ \mathbf{U}_c \end{bmatrix} = \begin{bmatrix} \mathbf{R}_f^g + \mathbf{R}_f^b \\ \mathbf{R}_c^g + \mathbf{R}_c^b \end{bmatrix}$$

thus leading to :

$$\mathbf{K}_{ff}\mathbf{U}_f = \mathbf{R}_f^g + \mathbf{R}_f^b - \mathbf{K}_{fc}\mathbf{U}_c \tag{30}$$

In the case of a linear tetrahedron finite element, $\mathbf{K}_{fc}\mathbf{U}_c$ is non zero only for free nodes that are neighbors to fixed nodes. In the remainder, we used simplified notations by dropping the subscript f for the stiffness matrix and displacement vector and by gathering all applied nodes into a single vector :

$$\mathbf{K}\mathbf{U} = \mathbf{R} \tag{31}$$

To get the dynamic law of motion, the work of inertial forces $-\frac{1}{2}\dot{\mathbf{U}}^T\mathbf{M}\dot{\mathbf{U}}$ should be added to the work of body forces. By adding the work of damping forces, the following classical equation is obtained :

$$\mathbf{M}\ddot{\mathbf{U}} + \mathbf{C}\dot{\mathbf{U}} + \mathbf{K}\mathbf{U} = \mathbf{R} \tag{32}$$

where $\mathbf{C}$ is the damping matrix. In general, we assume that $\mathbf{C}$ follows Rayleigh damping :

$$\mathbf{C} = \gamma_1 \mathbf{M} + \gamma_2 \mathbf{K} \tag{33}$$

This assumption is important for performing modal analysis but also for ensuring that the damping matrix, as the stiffness matrix, is also sparse.

4.11 Solution of equilibrium equations

The static equilibrium given by equation 31 is a linear system of equations with a symmetric positive definite stiffness matrix. Since this matrix is sparse, the classical method to solve this equation is to use the conjugated gradient algorithm [91].
More precisely, when solving the complete system $\mathbf{KU} = \mathbf{R}$, we perform the following steps :

- **Node Renumbering** by using the reverse cutting McKee algorithm [91] in order to decrease the bandwidth of the stiffness matrix.
- **Matrix Preconditioning** based on Cholesky factorisation or incomplete LU decomposition [91].
- **Application of the Conjugated Gradient Algorithm** for solving the linear system of equation. We rely on the *Matrix Template Library* [67] for an efficient implementation of these algorithms in C++. When the stiffness matrix is poorly conditioned, for instance for nearly incompressible materials, it is possible that the conjugated gradient algorithm fails. In which case, we resort to using direct methods for solving the system of equation, such as Gauss pivoting [91].

Despite optimizing the bandwidth and the condition number of the stiffness matrix, the time required for solving the static equation is still too large for real-time computation. For instance, with a liver model composed of a mesh consisting of 1313 vertices, the solution of the linear system of size 3939×3939 requires $9s$ on a PC Pentium II (450 Mhz) with 140 iterations of the preconditioned conjugated gradient in order to reach an accuracy of $0.001mm$. Therefore, solving directly the static equation with the conjugated gradient algorithm does not satisfy the real-time constraints mentioned in section 1.3.2 since $T_c > T_{relaxation}$. As an alternative, we propose in the next sections, three soft tissue models that satisfy either hard or soft real-time constraints.

5 Quasi-static Precomputed Linear Elastic Model

5.1 Introduction

Since the complete solution of the static equilibrium equation is too computationally expensive for real-time constraint, a straightforward solution is to perform only few iterations of the conjugated gradient at each time step in order to increase the update rate. This approach, proposed by Baraff *et al.* [4] is well suited in the context of computer animation but is not applicable for surgery simulation where boundary conditions are constantly changing and are

formulated in terms of imposed displacements. Indeed, using a conjugated gradient method would require to modify the stiffness matrix frequently as well as its preconditioning which would considerably reduce its efficiency.
Instead, we propose a *quasi-static precomputed linear elastic model* [21] that is based on a simple concept which consists in partially inverting the stiffness matrix in a precomputation stage before the simulation.
This model has the following characteristics :

- It is computationally very efficient : the computation complexity during the simulation is proportional to the cube of the number of imposed displacements.
- Only the position of surface nodes is updated during the simulation. In fact, only the data structure of the triangulated surface corresponding to the shell of the tetrahedral mesh is needed online.
- During the simulation the reaction forces at the nodes where the virtual instruments collide are also computed.
- The model is quasi-static, i.e. it computes the static equilibrium position at each iteration.

However, it relies on the following hypotheses :

- the mesh topology is not modified during the simulation. Thus, no simulation of cutting or suturing can be performed on this model.
- The interaction with neighboring tissues or with instruments is translated into modified boundary conditions (displacements or forces) only on surface nodes but not on the boundary conditions of internal nodes.

Therefore the main limitation of this precomputed model comes from the first hypothesis which states that it is not suited for the simulation of tissue cutting.

5.2 Overview of the Algorithm

One important feature of the model consists in making a distinction between surface and interior nodes. Thus, for the sake of clarity, we decompose the displacement and load vectors as well as the stiffness matrix according to surface and interior nodes with the S and I subscripts :

$$\begin{bmatrix} \mathbf{K}_{ss} & \mathbf{K}_{si} \\ \mathbf{K}_{is} & \mathbf{K}_{ii} \end{bmatrix} \begin{bmatrix} \mathbf{U}_s \\ \mathbf{U}_i \end{bmatrix} = \begin{bmatrix} \mathbf{R}_s \\ \mathbf{R}_i \end{bmatrix}$$

It is important to note that only free vertices appear in this matrix as discussed in section 4.10.
The solution of static equation can be obtained by multiplying the compliance matrix [**G**], corresponding to the inverse of the stiffness matrix [**K**], with the load vector. This compliance matrix can also be decomposed into surface and interior nodes :

$$\begin{bmatrix} \mathbf{U}_s \\ \mathbf{U}_i \end{bmatrix} = \begin{bmatrix} \mathbf{G}_{ss} & \mathbf{G}_{si} \\ \mathbf{G}_{is} & \mathbf{G}_{ii} \end{bmatrix} \begin{bmatrix} \mathbf{R}_s \\ \mathbf{R}_i \end{bmatrix} \tag{34}$$

The load vector $\mathbf{R}_s$ that applies on free surface nodes can be decomposed into two parts. A first part $\mathbf{R}_s^0$, corresponds to loads that will not evolve during the

simulation for instance gravity forces (see 4.6), constant pressure forces (see 4.7), applied nodal forces (see 4.9) or the presence of a non-zero imposed displacement vertex in its neighborhood (see equation 30). The second part $\mathbf{R}_s^C$ corresponds to loads that are created by the contact of the soft tissue with surgical tools.
The principle of this soft tissue model, is to compute the surface node positions $\mathbf{U}_s$ directly from the contact loads $\mathbf{R}_s^C$ by multiplying this vector with the compliance matrix $\mathbf{G}_{ss}$:

$$\begin{aligned} \mathbf{U}_s &= \mathbf{G}_{ss}\mathbf{R}_s^C + \mathbf{U}_s^0 \\ \mathbf{U}_s^0 &= \mathbf{G}_{ss}\mathbf{R}_s^0 + \mathbf{G}_{si}\mathbf{R}_i \end{aligned} \tag{35}$$

Since the loads on interior nodes $\mathbf{R}_i$ do not evolve during the simulation, $\mathbf{U}_s^0$ is a displacement offset that is computed as the displacement of surface nodes when no contact loads are applied : $\mathbf{R}_s^C = \mathbf{0}$.
The goal of the precomputation stage is to compute the compliance matrix $[\mathbf{G}_{ss}]$.

5.3 Precomputation stage

5.3.1 Description of the Algorithm

In the remainder, we write $[\mathbf{G}_{ss}^{ij}]$ the 3×3 submatrix of $\mathbf{G}_{ss}$ associated to vertex i and j. More precisely, a force $\mathbf{R}_s^j$ applied on vertex j entails an additional displacement of vertex i equal to : $[\mathbf{G}_{ss}^{ij}]\mathbf{R}_s^j$.

Algorithm 1 Computation of the compliance matrix $\mathbf{G}_{ss}$

```
Set R_i = 0
for all Surface Vertex i do
    for all j such that 0 ≤ j ≤ 2 do
        Set R_s = 0
        Set to 1.0 the j^th component of the load R_s^i applied to vertex i
        Solve the static equilibrium equation KU = R
        for all Surface Vertex k do
            Store the computed displacement U_k of vertex k into the j^th column
            of matrix [G_ss^ki]
        end for
    end for
end for
```

The algorithm for computing the compliance matrix $\mathbf{G}_{ss}$ is described above as Algorithm 1. It consists in solving $3 \times N_s$ times the linear system of equations $\mathbf{KU} = \mathbf{R}$, where N_s is the number of surface vertices. Note that the size of the stiffness matrix $\mathbf{K}$ is $N = N_s + N_i$ whereas the size of the compliance matrix $\mathbf{G}_{ss}$ is $3N_s \times 3N_s$.
The solution of equation $\mathbf{KU} = \mathbf{R}$ is performed using the steps described in section 4.11 including node renumbering and matrix preconditioning. Since the rigidity matrix $\mathbf{K}$ is the same for all $3 \times N_s$ systems of equations, these two steps are performed only once, which significantly speeds-up the computation. Each time a linear system of equation is solved, the displacement of all surface

nodes U_s corresponds to a column of matrix $\mathbf{G}_{ss}$. The storage of matrix $\mathbf{G}_{ss}$, requires only $\frac{8\times 9(N_s)^2}{2}$ bytes (each element being stored as a double), since it is a symmetric matrix, as the inverse of a symmetric matrix.
The algorithm presented above can be slightly improved in the following way :

- Applying a unitary force successively along the X, Y and Z directions may cause a loss of accuracy in computing the compliance matrix, because the resulting displacement may be very large or very small depending on the size of the mesh. To obtain meaningful displacements, it is possible to apply a force f_{ref} and then divide the resulting displacement by f_{ref} to compute $\mathbf{G}_{ss}$. A good choice for f_{ref} is $\|[\mathbf{K}_{i,i}]\| * 0.1 * l$, where $[\mathbf{K}_{i,i}]$ is the block diagonal stiffness matrix of vertex i, and l is the estimated diameter of the object. This choice of force scale, produces displacements which are roughly equal to 10% of the diameter.

- It is sometimes necessary to obtain the displacement of some interior nodes during the simulation. This is the case for instance when vessels or tumors, located inside an organ, need to be displayed during the simulation. In this case, it is possible in the final loop of the algorithm (lines 7, 8 and 9 of Algorithm 1) to add these inside vertices to the list of surface vertices. Thus, it does not entail the solution of any additional system of equations, but only an additional storage requirement since the compliance matrix becomes a rectangular matrix of size $3N_s \times 3(N_s + N_i^\star)$ where $N_i^\star$ is the number of additional interior nodes.

This precomputation stage is quite computationally expensive and requires between a few minutes up to several hours depending on the number of the mesh vertices and the stiffness of the material. For instance, the liver model presented in Figure 24 is composed of 1394 vertices, 8347 edges and 6342 tetrahedra. Its triangulated surface is composed of 1224 triangles and 614 vertices which is enough to produce a smooth visual rendering. The Poisson ratio of the material is set to 0.45 while its Young Modulus is $E = 1000kPa$. In this case, the precomputation time required nearly 4 hours on a Pentium Pii 450 Mhz, while the compliance matrix is stored in a file of size 13 Mb.

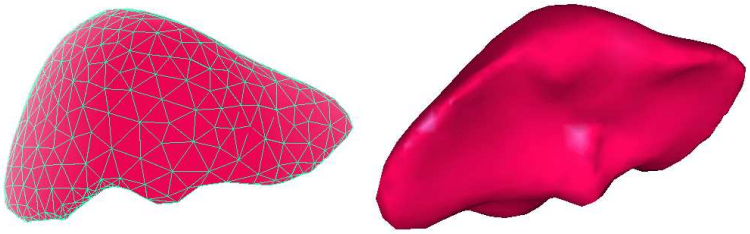
5.3.2 Other Methods for Computing the Compliance matrix

At least two alternative methods have been proposed in the literature to compute the compliance matrix $\mathbf{G}_{ss}$. The first one, proposed by Bro Nielsen *et al.* [13] is based on matrix condensation [68]. More precisely, the compliance matrix $\mathbf{G}_{ss}$ can be directly obtained from the inversion of the stiffness matrix $\mathbf{K}_{ii}$ of interior nodes. From equation 34, we can derive the following equations :

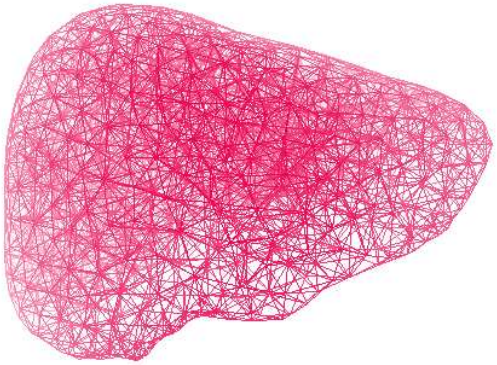
$$\begin{aligned} \mathbf{K}_{ii}\mathbf{U}_i &= \mathbf{R}_i - \mathbf{K}_{is}\mathbf{U}_s \\ \mathbf{K}_{ss}\mathbf{U}_s + \mathbf{K}_{si}\left(\mathbf{K}_{ii}^{-1}\mathbf{R}_i - \mathbf{K}_{ii}^{-1}\mathbf{K}_{is}\mathbf{U}_s\right) &= \mathbf{R}_s \\ \left(\mathbf{K}_{ss} - \mathbf{K}_{si}\mathbf{K}_{ii}^{-1}\mathbf{K}_{is}\right)\mathbf{U}_s &= \mathbf{R}_s - \mathbf{K}_{si}\mathbf{K}_{ii}^{-1}\mathbf{R}_i \end{aligned} \tag{36}$$

From equation 36, we can deduce the expression of the compliance matrix :

$$\mathbf{G}_{ss} = \left(\mathbf{K}_{ss}^\star\right)^{-1} = \left(\mathbf{K}_{ss} - \mathbf{K}_{si}\mathbf{K}_{ii}^{-1}\mathbf{K}_{is}\right)^{-1} \tag{37}$$



(a) Wireframe rendering (b) Gouraud shaded rendering



(c) Volumetric wireframe rendering

Figure 24: Visualization of a liver model with 1394 vertices and 6342 tetrahedra

Therefore, the computation of $\mathbf{G}_{ss}$ requires the inversion of two matrices : the first one of size $3N_i \times 3N_i$ and the second one of size $3N_s \times 3N_s$. This method has the disadvantage of requiring the additional storage of $9(N_i)^2$ numbers in double format, which in general is greater than the size of the compliance matrix : for large meshes, this method may become unpractical. Furthermore, this method is slightly more complex to implement whereas the method proposed in the previous section only requires to solve equation $\mathbf{KU} = \mathbf{R}$ with a sparse matrix $\mathbf{K}$. However, the condensation method is well suited when the rigidity matrix is very ill-conditioned (Poisson ratio very close to 0.5) in which case the preconditioned conjugated gradient algorithm may fail.

The second algorithm for computing the compliance matrix $\mathbf{G}_{ss}$ is to use the Boundary Element Method (BEM) [16] instead of the Finite Element Method (FEM). The algorithm proposed by James and Pai [52] creates the stiffness matrix $\mathbf{K}^{\star}_{ss}$ directly from the triangulated surface of the object.

The differences between BEM and FEM are well understood [50]. The main advantage of BEM techniques is that they do not require a volumetric tetrahedral mesh but only its triangulated surface. While there exist several free software [6] for automatically creating tetrahedral meshes from triangulated surfaces [99, 79, 53], having a control over the final number of vertices and the quality of tetrahedral elements is still an issue.

[6] A list of available software can be found at the following two URLs : http://www-users.informatik.rwth-aachen.de/~roberts/meshgeneration.html and http://www.andrew.cmu.edu/user/sowen/softsurv.html

On the other hand, BEM techniques have several disavantages over FEM. First, they make strong hypotheses about the nature of the elastic material : only homogeneous and isotropic linear elastic materials can be modelled. Second, the computation of the compliance matrix, and above all its diagonal elements, is difficult to implement and often numerically unstable because singular integrals must be evaluated over each triangle. The quality of the triangle geometry can influence the stability of this computation. Third, this method cannot compute the displacement of any interior point, which can be a limitation when the displacement of internal structures (vessels, tumors, ...) is needed. Finally, the BEM presented in [52] uses centroid collocation to compute the stiffness matrix. Thus, this matrix allows to compute the displacements of the centroids of all triangles but not the displacements of the triangulation vertices. Therefore, the mesh being deformed is not the original triangulated mesh but its dual mesh which is called a *simplex mesh* [31]. Mapping the displacements of triangle centroids into the displacements of vertices is not trivial since the duality between triangulation and simplex meshes is not a one-to-one mapping from the geometrical standpoint [31].
To conclude, the algorithm proposed by James *et al.* is more difficult to implement than our method and it is only suitable for simple material. However, when there is no software program for creating tetrahedral meshes from triangulations, this approach should be used.

5.4 On-line Computation

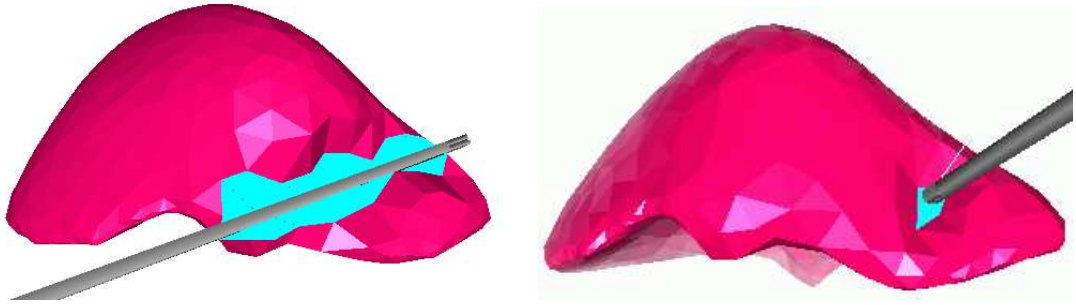
5.4.1 Data Structure

Before starting the simulation, the compliance matrix $\mathbf{G}_{ss}$, previously stored into a file as described in 5.3.1, is loaded into a specific data structure. Indeed, this data structure only describes the triangulated surface shell of the volumetric tetrahedral mesh with a list of surface vertices and a list of surface triangles. Note that the number of surface vertices is usually greater than N_s because some surface vertices have an imposed displacement. For display purposes, the triangulated data structure may contain additional information such as 2D or 3D texture coordinates as well as parameters describing the rendered material. Finally, the data structure contains a list of imposed displacements and applied nodal forces as a storage of boundary conditions.
For each free vertex of index i, an array of 3×3 matrices $[\mathbf{G}_{ss}^{ji}]$, for all $j \in \{0, \ldots, N_s - 1\}$, is stored inside the vertex data structure. These N_s matrices $[\mathbf{G}_{ss}^{ji}]$ allow to compute the displacement of all surface vertex j, once a force is applied on vertex i.
The data structure optimizes the computation time of deformation but at the cost of being less efficient in terms of memory requirement. Indeed, the compliance matrix $\mathbf{G}_{ss}$ is a symmetric matrix, but it is stored as a non-symmetric matrix in this data structure. To optimize memory at a small additional computational cost, one could alternatively store the symmetric matrix as a double array of 3×3 compliance matrices $[\mathbf{G}_{ss}^{ji}]$ which is filled only if $i < j$.

5.4.2 Algorithm description and collision processing

The sketch of the algorithm is given in the Algorithm 2 and includes two independent parts. The first part, between lines 1 and 8, consists in detecting and

computing the contact between the soft tissue model and each virtual surgical instrument. In Figure 25, we present an example of contact between a liver model and a tool. The collision detection algorithm [63] makes the assumption that the handle and the tool extremity can be approximated by a set of cylinders with rectangular section. Its efficiency depends on the availability of graphics cards since it relies on the OpenGL [110] library. Once a collision has been detected, the collided triangles must be moved such that the tissue model is no longer in contact with the surgical tool. This computation turns out to be quite complex since it not only depends on the tool position but also on its trajectory. The algorithm is described in [84]. The outcome of this computation is a list $l_{displacement}$ of imposed displacements that should apply on each vertex of the collided triangles.



(a) (b)

Figure 25: *Example of collision computation between the handle (a) and the extremity (b) of a surgical tool and a liver soft tissue model [84]. The position of triangles displayed in light gray have been displaced such that the tool is tangent to the liver surface.*

Algorithm 2 On-line computation of mesh deformation

Reset the list of imposed displacement $l_{displacement}$ to the empty list
Reset the list of applied forces l_{force} to the empty list
Reset the position of free surface vertices to their rest position + $\mathbf{U}_{ss}^0$
for all Surface Tools ST_i **do**
  if collision between the soft tissue model and ST_i **then**
    Add imposed displacement to the list $l_{displacement}$
  end if
end for
if $l_{displacement}$ is not empty **then**
  Compute the list of applied forces l_{force} from $l_{displacement}$
  for all Applied forces $\mathbf{F}_j^\star$ on vertex j in l_{force} **do**
    for all Free surface vertex k **do**
      Add to current position of vertex k, the displacement $[\mathbf{G}_{ss}^{kj}]\mathbf{F}_j^\star$
    end for
  end for
end if

5.4.3 Imposing displacements

The second part of Algorithm 2, between lines 9 and 16 computes the position of all surface vertices, given the list of imposed displacements.
The first task corresponding to line 10 consists in computing the set of forces $\{\mathbf{F}_j^\star\}$ that should be applied to each vertex j of $l_{displacement}$ in order to bring the displacement of these vertices to $\mathbf{U}_j^b$.
To be more didactic, we first consider that only one vertex displacement $\mathbf{U}_j^b$ is imposed on a vertex of index j. Without any collision with a surgical tool, this vertex has a displacement $\mathbf{U}_j^0$ under the application of the *normal* boundary conditions (gravity forces, pressure forces, ... described in section 4.9). Because the material is linear elastic, it follows the superposition principle : the displacements resulting from the application of two sets of nodal forces is the sum of the displacements resulting from the application of each set of forces. Thus the force $\mathbf{F}_j^\star$ to be computed is the force that should be applied on vertex j in order to create a displacement of that vertex equal to $\mathbf{U}_j^b - \mathbf{U}_j^0$. Because the quantity $[\mathbf{G}_{ss}^{jj}]\mathbf{F}_j^\star$ gives the additional displacement of vertex j resulting from the application of force $\mathbf{F}_j^\star$, the force $\mathbf{F}_j^\star$ is given by :

$$\mathbf{F}_j^\star = [\mathbf{G}_{ss}^{jj}]^{-1} \left(\mathbf{U}_j^b - \mathbf{U}_j^0\right)$$

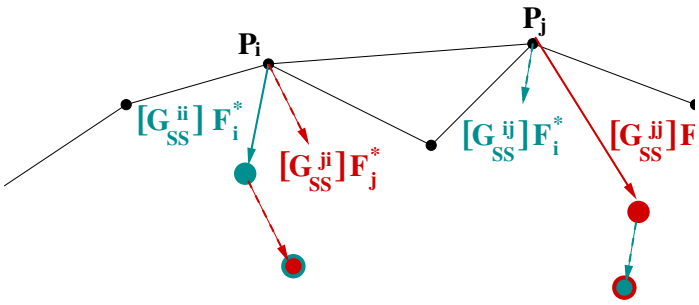


Figure 26: Principle of superposition when applying two forces $\mathbf{F}_i^\star$ and $\mathbf{F}_j^\star$ to the two nodes i and j

When the displacements of two vertices i and j are imposed, the problem is slighty more complex. Indeed the application of force $\mathbf{F}_i^\star$ on vertex i not only displaces vertex i of the amount $[\mathbf{G}_{ss}^{ii}]\mathbf{F}_i^\star$, but it also moves vertex j by the amount $[\mathbf{G}_{ss}^{ij}]\mathbf{F}_i^\star$ (see figure 26). Since $\mathbf{F}_j^\star$ also displaces vertex i of $[\mathbf{G}_{ss}^{ij}]\mathbf{F}_j^\star$, to compute the applied force, a 6×6 symmetric linear system of equations needs to be solved :

$$\begin{cases} \left[\mathbf{G}_{ss}^{ii}\right]\ \mathbf{F}_i^\star + \left[\mathbf{G}_{ss}^{ij}\right]\ \mathbf{F}_j^\star &= \mathbf{U}_i^b - \mathbf{U}_i^0 \\ \left[\mathbf{G}_{ss}^{ji}\right]\ \mathbf{F}_i^\star + \left[\mathbf{G}_{ss}^{jj}\right]\ \mathbf{F}_j^\star &= \mathbf{U}_j^b - \mathbf{U}_j^0 \end{cases}$$

Similarly, when the list of imposed displacements $l_{displacement}$ contains p elements, then a symmetric linear system of equations of size $3p \times 3p$ needs to be solved to find the set of nodal forces. If we use the set of indices i_j, $j \in [1 \dots p]$ to denote the set of vertices where a displacement $\mathbf{U}_{i_j}$ is imposed, then this linear system of equations can be written as :

$$\begin{bmatrix} \left[\mathbf{G}_{ss}^{i_1,i_1}\right] & \left[\mathbf{G}_{ss}^{i_1,i_2}\right] & \cdots & \left[\mathbf{G}_{ss}^{i_1,i_p}\right] \\ \left[\mathbf{G}_{ss}^{i_2,i_1}\right] & \left[\mathbf{G}_{ss}^{i_2,i_2}\right] & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \left[\mathbf{G}_{ss}^{i_p,i_1}\right] & \cdots & \cdots & \left[\mathbf{G}_{ss}^{i_p,i_p}\right] \end{bmatrix} \begin{bmatrix} \mathbf{F}_{i_1}^{\star} \\ \vdots \\ \vdots \\ \mathbf{F}_{i_p}^{\star} \end{bmatrix} = \begin{bmatrix} \mathbf{U}_{i_1}^{b} - \mathbf{U}_{i_1}^{0} \\ \vdots \\ \vdots \\ \mathbf{U}_{i_p}^{b} - \mathbf{U}_{i_p}^{0} \end{bmatrix} \tag{38}$$

In Figure 27, we show an example of a mesh where the same displacement is imposed on three vertices. In this particular case, the direction of computed forces departs strongly from the direction of the prescribed displacement.

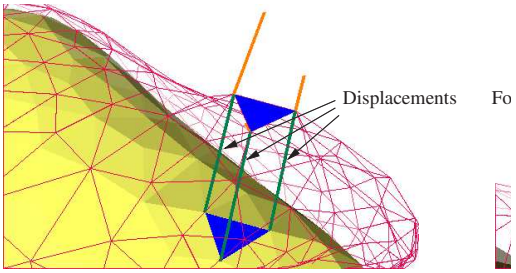


Figure 27: (Right) The same displacement is imposed on the three vertices of a triangle; (Left) From equation 38 we compute, the three forces that should be applied on these three vertices to move them of the given displacement.

5.4.4 Results

Once the set of nodal forces is computed, the additional displacement on all surface (and potentially internal) nodes are computed as described in lines 11 to 15 of Algorithm 2. The number of matrix-vector operations is $p \times N_s$ for p applied forces. In general, p, the number of vertices collided with the surgical tools, is small (from 3 to 20) when compared to N_s (see Figure 28). This is why we chosed to store the N_s array of compliance matrix $[\mathbf{G}_{ss}^{ji}]$ at vertex j, in order to optimize the inner loop (lines 12 to 14).

The computational efficiency of this quasi-static precomputed model on the liver mesh shown in Figure 24 is presented in table 3. These performances, measured on three different hardware platforms, correspond to the frequency update that can be achieved when running Algorithm 2 in a loop without any computation for visual and haptic rendering.

When applying one nodal force, corresponding to the execution of lines 12 to 14 in Algorithm 2, the computation time is nearly equal to 0.3 ms. The time required to compute the mesh deformation when applying p forces is strictly proportional to this value : $0.3 \times p$ ms.

When imposing p displacements, which is what occurs in practice in a surgical simulator, the additional computation is the solution of a $3p \times 3p$ linear symmetric system of equations. For $p = 1$, the overhead is very small and hardly perturbs the simulation frequency. However, for larger value of p, the overhead becomes dominant. For 20 vertices for instance, solving the system of equa-

Simulation Frequency (Liver model with 614 surface nodes)		Pentium PIII 600 MHz
Force applied on 1 node		3772 Hz
Force applied on 5 nodes		754 Hz
Force applied on 10 nodes		377 Hz
Force applied on 20 nodes		188 Hz
Imposed displacements on	1 node	3759 Hz
	5 nodes	561 Hz
	10 nodes	185 Hz
	20 nodes	40 Hz

Table 3: Computation efficiency of quasi-static precomputed linear elastic model for different boundary conditions : either when applying nodal forces or when imposing displacements.

tions of size 60×60 is 3 times more costly than computing the $20 * 614 = 1280$ matrix-vector products and additions.

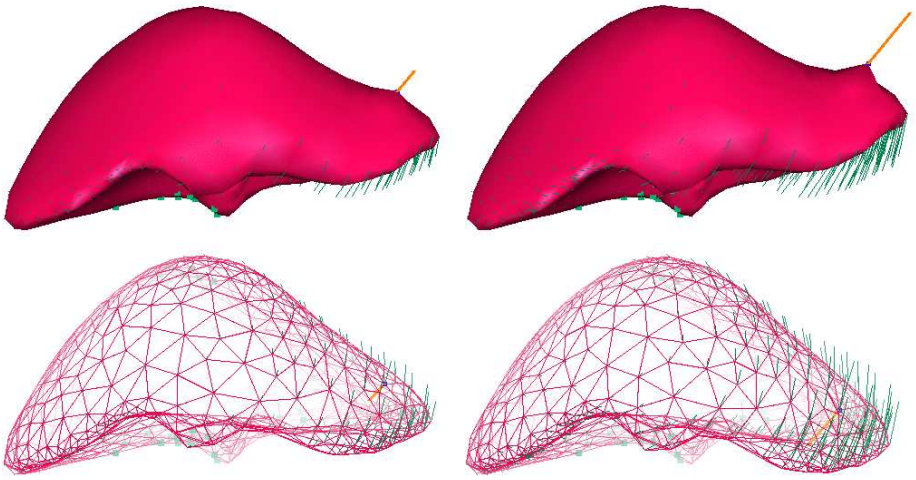


Figure 28: Liver deformation based on a linear elastic pre-computed model [22]. Solid lines indicate the imposed displacements.

5.4.5 Discussion

As a whole, the proposed method is "very efficient", since it allows real-time visual rendering even for large meshes. When the material is soft enough and when the number of collided vertices remains small (typically less than 15), this model can also be compatible with real-time haptic rendering. In fact, it is one of the few algorithms which are suitable for the first software architecture described in section 1.3.2 (see also Figure 6 (a)) consisting of one synchronous loop including visual and haptic rendering. Furthermore, our approach has one major advantage for haptic rendering computation : it already provides the nodal reaction forces through the algorithm described in 5.4.3. Indeed the set of forces $\mathbf{F}^{\star}_{i_j}$, corresponds to the set of physical forces that have been applied on each node of index i_j in order to deform the soft tissue model : thus, $-\mathbf{F}^{\star}_{i_j}$ corresponds to the nodal reaction force. From this set of forces, one can easily compute the reaction force along the direction of the tool, as well as the torque at the extremity of the tool.
Using the terminology introduced in section 1.3.2, we can also state that the quasi-static precomputed linear elastic model has a *very low relaxation time* (or equivalently that it has a high speed of convergence). Indeed, each time algorithm 2 is run, the soft tissue is deformed to its static equilibrium position. Because, this algorithm can be run at a high frequency, as seen in table 3, this implies that the relaxation time is very low. In fact, for some soft tissue, this time is too low and degrades the visual realism of the simulation. This is the case, for instance, when the operator grasps and displaces some soft tissue and suddenly ceases the grasping. Because the model has no longer any displacements imposed on its surface, it returns in one iteration to its rest position, while in reality, it takes several milliseconds.

Algorithm 3 Additional part of algorithm 2 that adds a visco-elastic behavior controlled by delay parameter γ

```
for all Free surface vertex k do
  if k ∉ l_displacement then
    Let p_k be the position of vertex k after algorithm 2
    Let p_k^previous be the position of vertex k at the previous iteration.
    p_k ⇐ γ p_k + (1 − γ) p_k^previous
  end if
  p_k^previous ⇐ p_k
end for
```

To add some visco-elastic behavior, one can increase the relaxation time artificially by using a delay function. This approach is described in the Algorithm 3. For vertices which are not colliding with a surgical tool, the final vertex position is a weighted sum between the position computed by algorithm 2 and the vertex position at the previous iteration. The weight parameter $0 \leq \gamma \leq 1$ controls the damping of the material deformation: for $\gamma = 1$, the deformation is not damped (quasi-static motion) while for $\gamma = 0$, the motion is infinitely damped (no motion). Any intermediate value of γ modifies the relaxation time of the material. Note that this damping is not applied to vertices colliding with tools because the collision would otherwise appear visually unrealistic. The algorithm 3 assumes that the model has a damping matrix $\mathbf{C}$ which is propor-

tional to the identity matrix : more sophisticated hypotheses (but often more computationnaly intensive) could be proposed.

6 Dynamic Linear Elastic Model

In this section, we describe two different soft tissue models that are able to address with the limitation of the previous model : the simulation of tissue cutting. Using the terminology defined in section 1.3.2, these two methods can be qualified as "Explicit Iterative Methods" sharing the advantage of requiring a small computation time for each iteration but with the drawback of having a low speed of convergence.
The main difference between these two models is that the first can model the visco-elastic behavior of the soft tissue properly whereas the second does not require the evaluation of any time step and is unconditionally stable.
Finally, we propose in section 6.3 a *hybrid model* which combines any of the two previous models with the precomputed linear elastic model seen in section 5.

6.1 Tensor-Mass Model

6.1.1 Introduction

The *Tensor-Mass model* is based on the dynamic law of motion described in equation 32 :

$$\mathbf{M}\ddot{\mathbf{U}} + \mathbf{C}\dot{\mathbf{U}} + \mathbf{K}\mathbf{U} = \mathbf{R}$$

This second order differential equation couples the motion of tissue under the influence of inertia $\mathbf{M}\ddot{\mathbf{U}}$, of visco-elasticity $\mathbf{C}\dot{\mathbf{U}}$, elasticity $\mathbf{KU}$ and external loads $\mathbf{R}$.
The most efficient way to solve the equation above is by far to use modal analysis [6]. By making simple assumptions about the damping matrix $\mathbf{C}$, it is possible to simplify the above PDE into a small set of ordinary differential equations with an appropriate change of basis. The proper basis is given by the eigenvectors associated to the largest eigenvalues of the generalized eigenproblem $\mathbf{K}\phi = \omega^2\mathbf{M}\phi$.
However, the eigenproblem must be solved each time the rigidity matrix is modified. Therefore, this approach is not suitable for simulating tissue cutting, since the computation cost to solve the eigenproblem is very high.
Instead, a classical method to solve equation 32, is to use integration methods : the time dimension is uniformly discretized with a time step Δt, and each term of that equation is supposed to be constant during each time interval. There is an important distinction between *implicit integration schemes* and *explicit integration schemes* depending whether the position of the model at time $t+\Delta t$ requires the solution or not of a global linear systems of equations (see also the discussion in section 1.3.2).
Implicit schemes are *unconditionally stable* which allows the use of large time steps. In structural analysis, the Houbolt method [47, 6] and the Newmark method [77, 6] are the most commonly used. However, these schemes require either to inverse a sparse matrix or to solve at each iteration a linear system of equations. Considering the time required to solve such a linear system (a

few seconds for a small-size mesh), these implicit schemes cannot be used for real-time interaction.
Instead, we chose to use *explicit integration schemes* which have several nice properties (ease of implementation, low computational cost) compared to implicit schemes but with the drawback of being *conditionally stable* : the time step must be smaller than a critical time step $\Delta t_{critical}$. Therefore, smaller time step Δt must be used for explicit schemes which yields a larger relaxation time and a longer time for convergence.

6.1.2 Mass Matrix

Regarding the mass matrix, a common choice consists in replacing the symmetric positive definite matrix $\mathbf{M}$ with a diagonal matrix, where each diagonal element is the sum of all row elements in the original matrix : this lumped mass matrix is detailed in section 4.8.
In order to keep the time step Δt large enough during the simulation, we propose a further simplification of the mass matrix $\mathbf{M}$ by considering that the nodal mass is constant for all nodes, which makes $\mathbf{M}$ proportional to the identity matrix :

$$\mathbf{M} = m_0 \mathbf{I}_3$$

where m_0 is the average mass per node computed as the total mass of the tissue divided by the number of nodes in the initial mesh.
Indeed, the critical time step Δt of the iterative scheme is inversely proportional to the highest eigenvalue of the matrix $\mathbf{M}^{-1}\mathbf{K}$, while the speed of convergence is related to the ratio between the largest to the smallest eigenvalues of the same matrix, also called the *condition number* of that matrix.
From the equation of the nodal stiffness matrix $[\mathbf{K}_{i,i}]$, we can state that the nodal stiffness is proportional to the size (for instance the largest foot height) of all the tetrahedra surrounding each node :

$$[\mathbf{K}_{i,i}] = \sum_{\mathcal{T} \in \mathcal{S}(i)} \frac{1}{36V(\mathcal{T})} \left((\lambda_{\mathcal{T}} + \mu_{\mathcal{T}})(\mathbf{m}_i \otimes \mathbf{m}_i) + \mu_{\mathcal{T}} A_i^2 \mathbf{I}_3 \right)$$

Thus, the largest eigenvalue of $\mathbf{K}$ is determined by the largest tetrahedra while the condition number is given by the size ratio between the largest and smallest tetrahedra. On the other hand, when performing mass lumping, as in [12], the nodal mass of $\mathbf{M}^{-1}$ is inversely proportional to the volume of tetrahedra surrounding each node. Therefore, the power spectrum of $\mathbf{M}^{-1}\mathbf{K}$ largely differs from that of $\mathbf{K}$: the largest eigenvalue of $\mathbf{M}^{-1}\mathbf{K}$ now becomes related to the tetrahedron of smallest size while the condition number is related to the square ratio between the largest and smallest tetrahedra. These properties of $\mathbf{M}^{-1}\mathbf{K}$ have two consequences for the simulation of tissue cutting : both the speed of convergence and the time step Δt decrease as tetrahedra of small size are created.
By choosing a mass matrix proportional to the identity matrix, we keep the spectral properties of the rigidity matrix : the creation of small tetrahedra does not entail any decrease of the time step and limits the decrease of the speed of convergence. However, this choice is a gross approximation of physics since the total mass of the tissue increases as the number of elements increases. As claimed in section 3.1.5, we prefer to satisfy real-time constraints of the simulation (by

keeping a large value of Δt) at the expense of coarse approximations of the tissue dynamic behavior.

6.1.3 Numerical Integration

Several explicit iterative schemes can be proposed from equation 32 depending on the choice of damping matrix and discretization of time derivatives. Below, we propose three explicit schemes that are of interest in the context of surgery simulation. In the remainder, we write ${}^t\mathbf{U}$ the displacement vector at time t.

Euler Method This method uses central finite differences to estimate acceleration but right finite difference to estimate speed. Furthermore, sophisticated damping matrix such as Rayleigh damping can be employed in this scheme :

$$\frac{m_0}{\Delta t^2}({}^{t-\Delta t}\mathbf{U} - 2{}^t\mathbf{U} + {}^{t+\Delta t}\mathbf{U}) + \frac{1}{\Delta t}(\gamma_1 m_0 \mathbf{I}_3 + \gamma_2 \mathbf{K})({}^t\mathbf{U} - {}^{t-\Delta t}\mathbf{U}) + \mathbf{K}{}^t\mathbf{U} = {}^t\mathbf{R}$$

The displacement at time $t + \Delta t$ can be computed through the recurrent equation :

$${}^{t+\Delta t}\mathbf{U} = {}^t\mathbf{U} + (1 - \Delta t\gamma_1)\left({}^t\mathbf{U} - {}^{t-\Delta t}\mathbf{U}\right) - \mathbf{K}\left(\frac{\Delta t^2}{m_0}{}^t\mathbf{U} + \frac{\gamma_2 \Delta t}{m_0}({}^t\mathbf{U} - {}^{t-\Delta t}\mathbf{U})\right) + \frac{\Delta t^2}{m_0}{}^t\mathbf{R}$$

Euler Method with central finite difference In this case, central finite differences are used to estimate both acceleration and speed, while constant damping is used $\gamma_2 = 0$:

$$\frac{m_0}{\Delta t^2}({}^{t-\Delta t}\mathbf{U} - 2{}^t\mathbf{U} + {}^{t+\Delta t}\mathbf{U}) + \frac{\gamma_1 m_0}{2\Delta t}({}^{t+\Delta t}\mathbf{U} - {}^{t-\Delta t}\mathbf{U}) + \mathbf{K}{}^t\mathbf{U} = {}^t\mathbf{R}$$

which leads to the following update equation :

$${}^{t+\Delta t}\mathbf{U} = {}^t\mathbf{U} + \frac{2 - \gamma_1 \Delta t}{2 + \gamma_1 \Delta t}({}^t\mathbf{U} - {}^{t-\Delta t}\mathbf{U}) - \frac{2\Delta t^2}{m_0(2 + \gamma_1 \Delta t)}(\mathbf{K}{}^t\mathbf{U} - {}^t\mathbf{R}) \qquad (39)$$

Runge-Kutta Method of order 4 The Runge-Kutta method [85] is an integration method of fourth order of accuracy, but which requires four evaluations of the Euler recurrent equation. To describe this method, it is necessary to write the original equation as a first order differential equation :

$$\frac{d}{dt}\begin{bmatrix}\dot{\mathbf{U}}\\ \mathbf{U}\end{bmatrix} = \begin{bmatrix}\ddot{\mathbf{U}}\\ \dot{\mathbf{U}}\end{bmatrix} = \begin{bmatrix}-\frac{\mathbf{C}}{m_0} & -\frac{\mathbf{K}}{m_0}\\ 1 & 0\end{bmatrix}\begin{bmatrix}\dot{\mathbf{U}}\\ \mathbf{U}\end{bmatrix} + \begin{bmatrix}\frac{\mathbf{R}}{m_0}\\ 0\end{bmatrix}$$

Now, the state of a soft tissue model at time t is described by two vectors : displacement vector ${}^t\mathbf{U}$ and the velocity vector ${}^t\dot{\mathbf{U}}$. Applying the simple Euler method on this system gives the following relation :

$$\begin{bmatrix}{}^{t+\Delta t}\dot{\mathbf{U}}\\ {}^{t+\Delta t}\mathbf{U}\end{bmatrix} - \begin{bmatrix}{}^t\dot{\mathbf{U}}\\ {}^t\mathbf{U}\end{bmatrix} = \Delta t\begin{bmatrix}\frac{1}{m_0}(-\mathbf{C}{}^t\dot{\mathbf{U}} - \mathbf{K}{}^t\mathbf{U} + {}^t\mathbf{R})\\ {}^t\dot{\mathbf{U}}\end{bmatrix} = \begin{bmatrix}\delta v({}^t\mathbf{U}, {}^t\dot{\mathbf{U}})\\ \delta u({}^t\mathbf{U}, {}^t\dot{\mathbf{U}})\end{bmatrix}$$

The fourth order Runge-Kutta method, requires to compute the following eight incremental displacement and velocity vectors :

$$\begin{aligned}
\delta v_1 &= \delta v({}^t\mathbf{U}, {}^t\dot{\mathbf{U}}) & \delta u_1 &= \delta u({}^t\mathbf{U}, {}^t\dot{\mathbf{U}}) \\
\delta v_2 &= \delta v({}^t\mathbf{U} + \frac{\delta u_1}{2}, {}^t\dot{\mathbf{U}} + \frac{\delta v_1}{2}) & \delta u_2 &= \delta u({}^t\mathbf{U} + \tfrac{\delta u_1}{2}, {}^t\dot{\mathbf{U}} + \tfrac{\delta v_1}{2}) \\
\delta v_3 &= \delta v({}^t\mathbf{U} + \frac{\delta u_2}{2}, {}^t\dot{\mathbf{U}} + \frac{\delta v_2}{2}) & \delta u_3 &= \delta u({}^t\mathbf{U} + \tfrac{\delta u_2}{2}, {}^t\dot{\mathbf{U}} + \tfrac{\delta v_2}{2}) \\
\delta v_4 &= \delta v({}^t\mathbf{U} + \frac{\delta u_3}{2}, {}^t\dot{\mathbf{U}} + \frac{\delta v_3}{2}) & \delta u_4 &= \delta u({}^t\mathbf{U} + \tfrac{\delta u_3}{2}, {}^t\dot{\mathbf{U}} + \tfrac{\delta v_3}{2})
\end{aligned}$$

Finally, the velocity and displacement for the next time step are given by the following equation :

$$\begin{bmatrix} {}^{t+\Delta t}\dot{\mathbf{U}} \\ {}^{t+\Delta t}\mathbf{U} \end{bmatrix} = \begin{bmatrix} {}^t\dot{\mathbf{U}} \\ {}^t\mathbf{U} \end{bmatrix} + \frac{1}{6}\begin{bmatrix} \delta v_1 \\ \delta u_1 \end{bmatrix} + \frac{1}{3}\begin{bmatrix} \delta v_2 \\ \delta u_2 \end{bmatrix} + \frac{1}{3}\begin{bmatrix} \delta v_3 \\ \delta u_3 \end{bmatrix} + \frac{1}{6}\begin{bmatrix} \delta v_4 \\ \delta u_4 \end{bmatrix}$$

Comparison between the three methods We summarized in table 4 the properties of the three methods described above. Three qualitative criteria were proposed to outline the advantages and drawbacks of each method. In terms of computation time required to update the position of a model, the first two Euler methods are equivalent while the Runge Kutta method is at least four times slower. As far as damping is concerned, only the first Euler method allows to use Rayleigh damping while the two other methods can only use diagonal damping matrices. Having a non-diagonal damping matrix helps in keeping a continous field of velocity throughout the model which improves the visual realism of the simulation. Finally, the Runge-Kutta method is more stable than the Euler Method and our experience showed that a tenfold increase of the time step can be observed in the former case. The Euler method with central finite differences allows larger time steps than the Euler Method because the velocity computation leaps over position computation by one time step.

	Euler Method	Euler Central Finite Differences	Runge-Kutta Method
Computation time	low	low	high
Damping	Rayleigh	Basic	Basic
Time Step	small	medium	high

Table 4: Comparison between three explicit integration methods for soft tissue modeling.

6.1.4 Data Structure

With explicit schemes, the update of the mesh position can be performed locally, at the vertex level, without creating any global matrix. Indeed, for each free vertex of index i, we can take advantage of the sparse nature of the rigidity matrix $\mathbf{K}$, in order to compute the matrix-vector product $\mathbf{KU}$. More precisely, from equation 23, it is clear that the off-diagonal stiffness matrices $[K_{i,j}]$ are non-null matrices only when there is an edge connecting vertices i and j in the

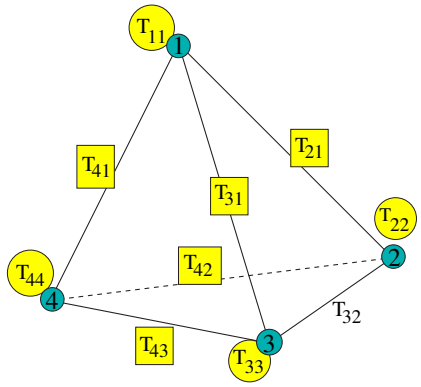


Figure 29: Representation of the data structure of a Tensor-Mass model. The 3×3 rigidity matrices are stored at each edge and each vertex. The symmetry of the rigidity matrix enables to store only one tensor per edge.

tetrahedral mesh. Therefore, only the set $\mathcal{N}(i)$ of vertices connected to vertex i by an edge is involved when computing the elastic force $\mathbf{F}_i$ applied on vertex i. For instance, the update equation 39 can be computed for a vertex i as :

$$
\begin{aligned}
{}^{t+\Delta t}\mathbf{u}_i = {}^{t}\mathbf{u}_i + \frac{2-\gamma_1\Delta t}{2+\gamma_1\Delta t}({}^{t}\mathbf{u}_i - {}^{t-\Delta t}\mathbf{u}_i) - \\
\frac{2\Delta t^2}{m_0(2+\gamma_1\Delta t)}\left(\sum_{j\in\mathcal{N}(i)}[\mathbf{K}_{i,j}]{}^{t}\mathbf{u}_j + [\mathbf{K}_{i,i}]{}^{t}\mathbf{u}_i - {}^{t}\mathbf{R}_i\right)
\end{aligned}
$$

The data structure that is suitable for performing this computation follows the data structure required for storing a tetrahedral mesh. The basic structure consists in a double-linked list of vertices, edges and tetrahedra. For each vertex, we store its current position ${}^{t}\mathbf{q}_i$, its rest position $\mathbf{p}_i$ and the symmetric tensor $[\mathbf{K}_{i,i}]$. For each edge, we store its two adjacent vertices (vertex i and vertex j) as well as the tensor $[\mathbf{K}_{i,j}]$, as sketched in Figure (29). We therefore take advantage of the symmetric nature of the stiffness matrix by storing the off-diagonal rigidity matrix only once.
Finally for each tetrahedron, we store its four vertices and its six edges as well as the Lamé coefficients λ_i, μ_i, the area vectors $\mathbf{m}_i$ and if required the direction of anisotropy $\mathbf{a}_0$.

6.1.5 Cutting and Refinement Algorithms

One of the basic tasks in surgery simulation consists in cutting and tearing soft tissue. With the dynamic linear elastic model, these tasks can be achieved efficiently.
To perform an hepatectomy (partial resection of the liver), the use of scalpel instruments is not appropriate because of the important vascularization of the liver. Instead, surgeons usually proceed with a set of pliers that smash hepatic cells or with a cavitron device that destroys the hepatic parenchyma with ultrasound energy : in both cases, the resection is performed by removing soft tissue.

It is therefore important to simulate the removal of bits of soft tissue located at the vicinity of a surgical tool. To perform this simulation, two basic meshing techniques must be implemented : removal of tetrahedra and local refinement. At first sight, removing a single tetrahedron from a tetrahedral mesh is straightforward. However, in order to obtain a visually realistic simulation, one should avoid to produce isolated or self-intersecting tetrahedra or even tetrahedra connected through a single vertex. A proper way to keep "visually appealing" meshes is to constrain the mesh to be a *manifold* mesh in addition to being a *conformal* mesh. Indeed, in a manifold mesh, the shell of a vertex located on the mesh surface is homeomorphic a half-sphere (the shell is a sphere for interior vertices) which allows to define unambiguously a surface normal at that vertex. However, by adding this topological constraint, even removing a single tetrahedron is not straightforward as discussed in [38]. The detailed description of the topological issues relevant to the operation of tetrahedron removal falls outside the scope of this chapter; instead we present briefly the algorithms related to the computation of soft tissue deformation.

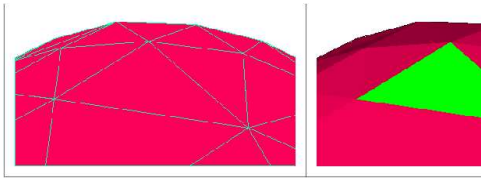


Figure 30: To remove the tetrahedron whose external triangle has been selected (dark gray), it is necessary to update the local rigidity matrices stored at the vertices and edges of that tetrahedron.

Once a collision between a surgical tool and a set of tetrahedra has been detected, each tetrahedron of the set is removed one after the other. After updating the topological structure of the mesh, the local vertex and edge stiffness matrices must also be updated (see Figure 30). When removing tetrahedron $\mathcal{T}$, its 6 edge tensors $[\mathbf{B}_{i,j}^{\mathcal{T}}]$ and 4 vertex tensors $[\mathbf{B}_{i,i}^{\mathcal{T}}]$ are computed based on equations 18 and 19 and are substracted from the current edge and vertex local rigidity matrices :

$$[\mathbf{K}_{i,i}] = [\mathbf{K}_{i,i}] - [\mathbf{B}_{ii}^{\mathcal{T}}] \qquad [\mathbf{K}_{i,j}] = [\mathbf{K}_{i,j}] - [\mathbf{B}_{ij}^{\mathcal{T}}]$$

These ten local operations are performed efficiently because of the specific data structure associated with a tetrahedron.
The second meshing technique, local refinement, can be used in two cases. First, it can be used offline (before the simulation), to increase the mesh resolution at places of high curvature or near structures of interest (tumors, gall blader,...). Second, it is often necessary to refine the mesh locally during the removal of soft tissue when the tetrahedra to be removed are too large. In the former case, sophisticated meshing techniques can be employed while in the latter case, real-time constraints allow the application of only basic refinement algorithms. An example of such a basic algorithm consists in adding a vertex at the middle of an edge and then splitting all tetrahedra adjacent to that edge into two tetrahedra (see Figure 31). In this case, the edge and vertex tensors of all tetrahedra

adjacent to that edge are first removed and the contributions from all newly created tetrahedra are then added. A more sophisticated refinement algorithm can be found in [38].

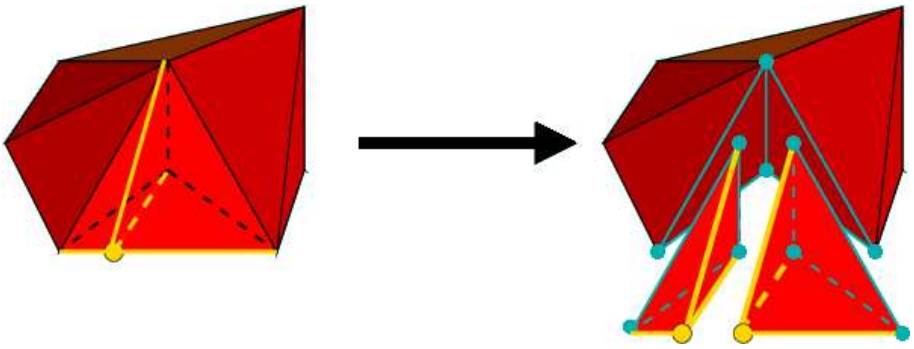


Figure 31: Local refinement of a tetrahedral mesh. An edge is split into two edges by inserting a vertex. The rigidity matrices must be updated for vertices and edges that already existed (drawn in dark grey) while these matrices must be computed for newly created vertices and edges (drawn in light grey).

The proper adjustment of stiffness matrices during the removal of soft tissue reinforces the visual realism of the simulation significantly: this is especially the case when the tissue is cut while being stretched. For instance in Figure 32, we show the deformation of a cylinder being cut : the cylinder is fixed at its upper part and is under the influence of gravity forces along its main axis.

6.1.6 Algorithm Description

Before describing the deformation algorithm for a Tensor-Mass model, we shortly describe the initialization stages in Algorithm 4. Once the vertex and edge stiffness matrices have been assembled, it is necessary to estimate a time step Δt that allow the stability of the iterative schemes described in section 6.1.3. Finding the critical time step (i.e. the highest possible time step) is actually a difficult task because of the lack of a closed-form expression. However, a practical approach is to estimate the critical time step as a product of an unknown constant with the time step given by the Courant-Friedrich-Levy condition [86] :

$$(\Delta t)_{Courant} = l_{max}\sqrt{\frac{\rho}{\lambda + 2\mu}}$$

Algorithm 5 presents the different loops required to update a Tensor-Mass model. Unlike the precomputed quasi-static model, it is not necessary to maintain an explicit list of vertices that are displaced by the collision with a surgical tool : it is sufficient (see line 7) to raise a flag stating that these vertices are not free vertices. A second important feature of this algorithm is the existence of a loop on the mesh edges in order to compute the matrix-vector products $\sum_{j\in\mathcal{N}(i)}[\mathbf{K}_{i,j}]^t\mathbf{u}_j$. This approach is more efficient than scanning iteratively the neighbors $\mathcal{N}(i)$ for each vertex i. When using the fourth order Runge-Kutta algorithm, the algorithm from lines 10 to 21 must be modified since it

Figure 32: Deformation of a cylinder subject to gravity forces : some tetrahedra are progressively being removed at its center leading to a separation into independent solids.

Algorithm 4 Matrix Assembly for the Tensor-Mass model performed before any simulation

for all Tetrahedron $\mathcal{T}$ **do**
  Compute the 4 area vectors $\mathbf{m}_i$
  for all Vertex i of $\mathcal{T}$ **do**
    Compute the local rigidity matrix $[\mathbf{B}_{ii}^{\mathcal{T}}]$
    $[\mathbf{K}_{i,i}] \Leftarrow [\mathbf{K}_{i,i}] + [\mathbf{B}_{ii}^{\mathcal{T}}]$
  end for
  for all Edge between vertices i and j of $\mathcal{T}$ **do**
    Compute the local rigidity matrix $[\mathbf{B}_{ij}^{\mathcal{T}}]$
    $[\mathbf{K}_{i,j}] \Leftarrow [\mathbf{K}_{i,j}] + [\mathbf{B}_{ij}^{\mathcal{T}}]$
  end for
end for
Estimate time step Δt.

is then necessary to scan four times the edges and vertices of the mesh. For the Euler method, only lines 11 and 12 must be modified in order to compute $\mathbf{K}\left(\frac{\Delta t^2}{m_0}{}^t\mathbf{U} + \frac{\gamma_2 \Delta t}{m_0}({}^t\mathbf{U} - {}^{t-\Delta t}\mathbf{U})\right)$ instead of $\mathbf{K}{}^t\mathbf{U}$.

Algorithm 5 On-line computation of Tensor-Mass model

```
1: for all Surface Tools ST_i do
2:   if collision between the soft tissue model and ST_i then
3:     if ST_i represent a cavitron device then
4:       Eventually refine locally the mesh near the collision
5:       Remove tetrahedra located near the extremity of ST_i
6:     end if
7:     Impose displacements on vertices near the contact zone and raise a flag
       on these vertices
8:   end if
9: end for
10: for all edge e connecting vertex i and j do
11:   add elastic force [K_{i,j}]^t u_i to vertex i
12:   add elastic force [K_{i,j}]^T {}^t u_j to vertex j
13: end for
14: for all vertex i do
15:   if vertex i is free (flag not raised) then
16:     compute elastic force [K_{i,i}]^t u_i
17:     update vertex position {}^t p_i based on one of the three iterative schemes
        described in 6.1.3
18:   else
19:     reset flag
20:   end if
21: end for
```

6.1.7 Comparison between Spring-mass and Tensor-mass models

We have used the word "Tensor-Mass model" to designate a finite-element model based on Newtonian dynamics and discretized with an explicit scheme. This word has been chosen in order to stress the similarity between a "Tensor-Mass model" and a "spring-mass model". In particular, it is the purpose of this section to oppose to the widely spread belief stating that "finite element models are slower and more complex to implement than spring-mass models".
A spring-mass model [4] consists of a set of masses and a set of springs connecting these masses. The force applied to a point $\mathbf{p}_i$ in a spring-mass system, is given by the relation:

$$\mathbf{F}_i = \sum_{j \in \mathcal{N}(i)} k_{ij}(\|\mathbf{p}_i\mathbf{p}_j\| - l_{ij}^0)\frac{\mathbf{p}_i\mathbf{p}_j}{\|\mathbf{p}_i\mathbf{p}_j\|} \tag{40}$$

where k_{ij} is the stiffness coefficient between vertices i and j, l_{ij}^0 is the length at rest.
Similarly, on a Tensor-Mass model, the elastic force applied on vertex i is given by :

$$\mathbf{F}_i = [\mathbf{K}_{i,i}]\mathbf{u}_i + \sum_{j \in \mathcal{N}(i)} [\mathbf{K}_{i,j}]\mathbf{u}_j \tag{41}$$

By comparing equations 40 and 41, it is clear that both dynamic models have the same computational complexity which is linear in the number of edges. In practice, we have observed a slight computational advantage for the Tensor-Mass model, mostly because it does not include any square root evaluation.
However, both approaches differ substantially in terms of biomechanical modeling. Spring-mass systems constitute a discrete representation of an object and their behavior strongly depends on the topology of the spring network. Adding or removing a spring may change the elastic behavior of the whole system drastically. Conversely, a finite element model is a continuous representation of the object and its behavior is independent of the mesh topology (it mostly depends on the mesh resolution). This is an advantage when mesh cutting is performed since it produces continuous and natural deformations.
Because all biomechanical data related to biological soft tissue are formulated as parameters found in continuum mechanics (such as Young's modulus or Poisson coefficients), it is *a priori* difficult to model realistic soft tissue deformations with a spring-mass system. However, several authors [66, 34] have developed genetic or simulated annealing algorithms to identify spring parameters (stiffness and damping) from a set of known deformations of an object.
Finally as previously mentioned, the Tensor-Mass model is only valid for small displacements. This model is invariant under the application of a global translation, but if a global rotation is applied to the rest shape $\mathcal{M}_{rest}$, then the forces applied to all vertices will not be null. On the opposite, a spring-mass model under the same displacement would not deform, since the length of the springs are preserved under a rigid transformation. The difference between these three soft tissue models is summarized in Table 5.

	Pre-computed	Tensor-Mass	Spring-Mass
Computational efficiency	+++	+	+
Biomechanical Realism	+	+	-
Cutting simulation	-	++	+
Large displacements	-	-	+

Table 5: Comparison between the three soft tissue models: pre-computed quasi-static, Tensor-Mass and spring-mass models.

6.2 Relaxation-based elastic models

6.2.1 Introduction

In this section, we introduce an alternative algorithm to the Tensor-Mass model. This algorithm is based on Gauss-Seidel relaxation and has the following properties :

- Its iterative scheme is unconditionally stable. It does not require the estimation of any critical time step.
- The relaxation algorithm is fairly efficient (small computation time required for one iteration) but it is slightly less efficient than a Tensor-Mass model.

- The algorithm is based on static equilibrium equations whereas Tensor-Mass models are based on the dynamic law of motion.
- The position of each vertex is updated asynchronously, one vertex after the other.

However, when compared to Tensor-Mass models, relaxation-based elastic models have two drawbacks. First, their implementation requires the following property for the mesh data structure : each vertex should be able to access efficiently its adjacent edges. This topological "vertex-edge" relationship can be stored in two ways inside a data structure. In a first approach, a list of edges can be stored explicitly at each vertex. After removing or adding tetrahedra, the edge list must be updated for all vertices belonging to these tetrahedra. To achieve this update, each edge must have a list of adjacent tetrahedra which should also be explicitly updated upon the removal or addition of tetrahedra.
In a second approach, the list of edges adjacent to a vertex is recovered through the knowledge of a single tetrahedron adjacent to this vertex. This approach is only applicable if we constrain the tetrahedral mesh to be a manifold mesh (see [38] for more details). Indeed, in such case, the neighborhood of a vertex is homeomorphic to a topological sphere or half-sphere. By marching around a vertex from a given tetrahedron, it is possible to obtain all tetrahedra adjacent to a given vertex and consequently the list of all adjacent edges. In this case as in the former case, we do store a list of adjacent edges for each vertex in order to avoid duplicating the search algorithm. However when a tetrahedron is removed or added, this topological list is reseted and the pointer to the adjacent tetrahedron is eventually updated.
The second drawback of relaxation algorithms is that they require in average 3 times more storage than the Tensor-Mass model. Indeed, in addition to the symmetric stifness matrix, a non-symmetric stiffness matrix must be stored.

6.2.2 Overview of the Algorithm

Following the notations of equation 41 the static problem $\mathbf{KU} = \mathbf{R}$ can be written at the level of each vertex i as :

$$[\mathbf{K}_{i,i}]\mathbf{u}_i + \sum_{j \in \mathcal{N}(i)} [\mathbf{K}_{i,j}]\mathbf{u}_j = \mathbf{R}_i \tag{42}$$

For relaxation algorithms, the displacement of a vertex $\mathbf{u}_i$ is updated independently from other vertices. Therefore, the notation ${}^{t+\Delta t}\mathbf{u}_i$ to describe the position of vertex i at the next time step cannot be used, since formally there is no temporal evolution (and no temporal variable t) in relaxation algorithms. Thus, we note ${}^{+}\mathbf{u}_i$ the next position of vertex i and $\mathbf{u}_i$ its current position.
The principle of relaxation algorithms is quite straightforward : each vertex is moved in order to locally solve equation 42. Thus the displacement ${}^{+}\mathbf{u}_i$ is given by :

$${}^{+}\mathbf{u}_i = - \sum_{j \in \mathcal{N}(i)} [\mathbf{K}_{i,i}]^{-1} [\mathbf{K}_{i,j}]\mathbf{u}_j + [\mathbf{K}_{i,i}]^{-1}\mathbf{R}_i \tag{43}$$

This is equivalent to minimizing the total mechanical energy by successively optimizing each variable $\mathbf{u}_i$. It is therefore similar to the Iterative Conditional Mode (ICM) algorithm [9] used in statistical analysis.

If all displacements $\{\mathbf{u}_i\}$ are successively updated according to equation 42, then this method is equivalent to the Gauss-Seidel relaxation method [91]. More precisely, we can decompose the stiffness matrix $\mathbf{K}$ as the sum of three terms :$\mathbf{K}_D$ a 3×3 block diagonal matrix, $\mathbf{K}_C$ the lower triangle matrix of $\mathbf{K}$ and $\mathbf{K}_C^T$ the upper triangle matrix of $\mathbf{K}$:

$$\mathbf{K} = \underbrace{\begin{bmatrix} [\mathbf{K}_{1,1}] & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & [\mathbf{K}_{2,2}] & \ddots & \mathbf{0} \\ \vdots & \ddots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & [\mathbf{K}_{N,N}] \end{bmatrix}}_{\mathbf{K}_D} + \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ [\mathbf{K}_{2,1}] & \mathbf{0} & \ddots & \mathbf{0} \\ \vdots & \ddots & \ddots & \vdots \\ [\mathbf{K}_{N,1}] & [\mathbf{K}_{N,2}] & \cdots & \mathbf{0} \end{bmatrix}}_{\mathbf{K}_C} + \mathbf{K}_C^T$$

With this notation, the Gauss-Seidel relaxation consists in the application of an iterative equation :

$$^{k+1}\mathbf{U} = (\mathbf{K}_D + \mathbf{K}_C)^{-1} \left(-\mathbf{K}_C^T \ ^k\mathbf{U} + \mathbf{R}\right) \tag{44}$$

where $^k\mathbf{U}$ is the displacement vector at iteration k.
To speed-up convergence, we use over-relaxation (known as the Simultaneous Over-Relaxation algorithm [91]) that consists in anticipating future correction with an overrelaxation parameter ω :

$$^{k+1}\mathbf{U} = (\mathbf{K}_D + \omega\mathbf{K}_C)^{-1} \left(-\omega\mathbf{K}_C^T \ ^k\mathbf{U} + (1-\omega)\mathbf{K}_D \ ^k\mathbf{U} + \omega\mathbf{R}\right) \tag{45}$$

This equation translates at the vertex level with the recursion :

$$^+\mathbf{u}_i = (1-\omega)\mathbf{u}_i - \omega \sum_{j \in \mathcal{N}(i)} [\mathbf{K}_{i,i}]^{-1} \ [\mathbf{K}_{i,j}]\mathbf{u}_j + \omega[\mathbf{K}_{i,i}]^{-1}\mathbf{R}_i \tag{46}$$

If $\omega = 1$, then the SOR algorithm is equivalent to the Gauss-Seidel relaxation. Convergence is guaranteed for values of ω comprised between 1 and 2, while fastest convergence is obtained for a critical value :

$$\omega_{optimal} = \frac{2}{1 + \sqrt{1 - \rho_{GS}}}$$

where ρ_{GS} is the spectral radius (the modulus of the largest eigenvalue) of the matrix $(\mathbf{K}_D + \omega\mathbf{K}_C)^{-1} \ \mathbf{K}_C^T$.
The overelaxation parameter ω controls the dynamics of the soft tissue model. With $\omega \equiv 2$, the model tends to overshoot around the solution whereas with $\omega \equiv 1$, the motion is very damped. In practise, we chose a value of $\omega = 1.2$ as a trade-off between these two behaviors.

6.2.3 Algorithm Description

The application of the SOR recursive equation 46 requires the computation of matrices $[\mathbf{K}_{i,i}]^{-1} \ [\mathbf{K}_{i,j}]$ and $[\mathbf{K}_{i,i}]^{-1}$. For speed-up purposes, these matrices are stored respectively at each vertex and edge. Because the matrix $\mathbf{K}_D^{-1}\mathbf{K}$ is no longer symmetric, at each edge linking vertices i and j, we store the two 3×3 matrices $[\mathbf{K}_{i,i}]^{-1} \ [\mathbf{K}_{i,j}]$ and $[\mathbf{K}_{j,j}]^{-1} \ [\mathbf{K}_{i,j}]^T$.

The algorithm of the relaxation-based elastic model is presented as Algorithm 6. A large part is dedicated to the update of these additional matrices each time a topological change of the mesh occurs. A flag is positionned at each vertex and edge in order to indicate whether matrices $[\mathbf{K}_{i,i}]^{-1}$ $[\mathbf{K}_{i,j}]$ and $[\mathbf{K}_{i,i}]^{-1}$ are up-to-date or not. This flag is raised each time a topological change takes place at a vertex or edge level and it is lowered once these matrices are updated.

Algorithm 6 On-line computation of the relaxation-based model

for all Surface Tools ST_i **do**
  if collision between the soft tissue model and ST_i **then**
    if ST_i represents a cavitron device **then**
      Possibly refine locally the mesh near the collision
      Remove tetrahedra located near the extremity of ST_i
    end if
    Impose displacements on vertices near the contact zone
  end if
end for
for all free vertex i **do**
  if flag raised at vertex i **then**
    compute and store $[\mathbf{K}_{i,i}]^{-1}$
    lower flag at vertex i
  end if
  $\mathbf{u}_i^\star \Leftarrow (1-\omega)\mathbf{u}_i + \omega[\mathbf{K}_{i,i}]^{-1}\mathbf{R}_i$
  for all edge e connecting vertex i and j **do**
    if flag raised at edge e **then**
      if flag raised at vertex j **then**
        compute and store $[\mathbf{K}_{j,j}]^{-1}$
        lower flag at vertex j
      end if
      compute and store $[\mathbf{K}_{i,i}]^{-1}$ $[\mathbf{K}_{i,j}]$ and $[\mathbf{K}_{j,j}]^{-1}$ $[\mathbf{K}_{i,j}]^T$
      lower flag at edge e
    end if
    $\mathbf{u}_i^\star \Leftarrow \mathbf{u}_i^\star - \omega[\mathbf{K}_{i,i}]^{-1}$ $[\mathbf{K}_{i,j}]$ $\mathbf{u}_j$
  end for
  $\mathbf{u}_i \Leftarrow \mathbf{u}_i^\star$
end for

6.3 Hybrid Models

6.3.1 Motivation

We have previously described two types of *linear elastic models* :

1. a *quasi-static* pre-computed elastic model which is computationally efficient but that does not allow any change of topology(cutting, tearing) (see section 5).

2. two *dynamic* elastic models (Tensor-Mass and relaxation based models) that have lower convergence speed but that allow topology changes (see

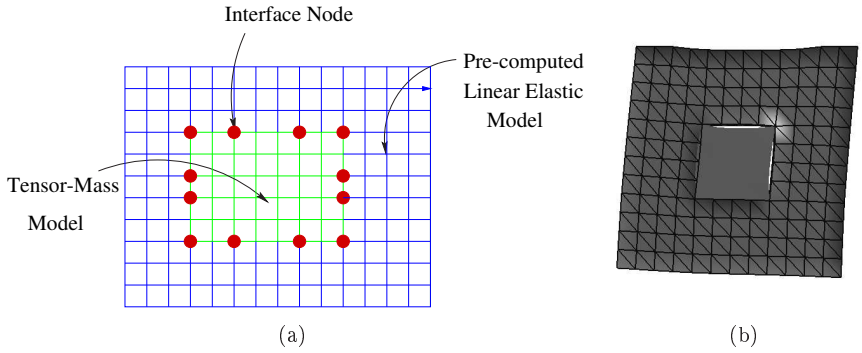


Figure 33: (a) Definition of the interface nodes in a hybrid elastic model; (b) Hybrid elastic model with eight interface nodes [23].

sections 6.1 and 6.2). In the remainder, we use Tensor-Mass models as the method for deforming

To combine these two approaches, we make a distinction between two types of anatomical structures that usually appear in a surgical simulation:

- anatomical structures which are the target of the surgical procedure. On these structures, tearing and cutting need to be simulated. In many cases, they correspond to pathological structures and only represent a small subset of the anatomy that needs to be visualized during the simulation.
- Anatomical structures which only need to be visualized or eventually deformed but which are not submitted to any surgical action.

Thus, in a hybrid model, we propose to model the former type of anatomical structures as Tensor-Mass models whereas the latter type of structures should be modeled as a pre-computed linear model. However, this method is only efficient if the number of Tensor-Mass elements is kept as low as possible.

6.3.2 Description

A hybrid elastic model $\mathcal{M}_{hybrid}$ is composed of two different types of elements : let $\mathcal{M}_{dynamic}$ be the set of Tensor-Mass elements and let $\mathcal{M}_{quasi-static}$ be the set of pre-computed linear elastic elements. The model $\mathcal{M}_{dynamic}$ is connected to $\mathcal{M}_{quasi-static}$ by a set of common vertices called *interface nodes*. These interface nodes define additional boundary conditions for each model. As seen in Figure 33, the two models may not be completely connected along their entire boundaries. In fact, a way to reduce the number of Tensor-Mass elements, is to associate a fine pre-computed elastic model with a coarse Tensor-Mass model. As shown in Figure 33-b , this incomplete interface causes some visual artifacts due to the non-continuity between two neighboring parts. However, if the interface zone between the two elastic models is not an important visual cue, a different mesh resolution can be used.
Since both linear elastic models follow the same physical law, their combination should behave exactly as a global linear elastic model. Thus, the additional

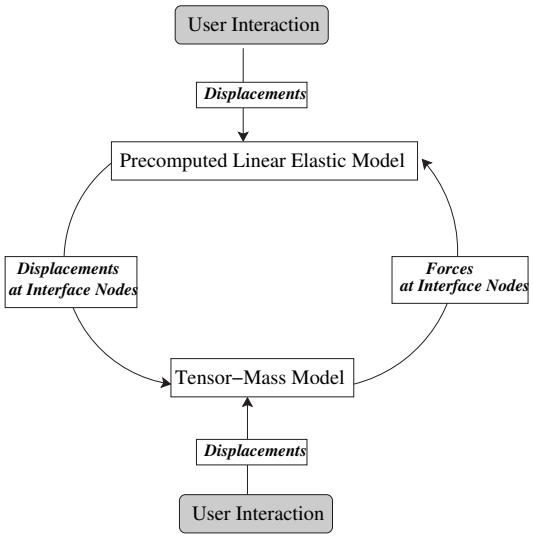


Figure 34: Interaction loop for a hybrid elastic model. Both models are updated alternatively while allowing for user interaction.

boundary conditions imposed at the interface nodes must be consistent with respoin terms of forces and displacements for both models.
Figure 34 summarizes the computation loop of a hybrid model. Since the precomputed model $\mathcal{M}_{quasi-static}$ is more efficient with force boundary conditions than with imposed displacements (see section 5.4.3), its update is based on forces applied at interface nodes by $\mathcal{M}_{\mathrm{dynamic}}$ but also on imposed displacements resulting from the contact with surgical tools. The applied forces originating from $\mathcal{M}_{\mathrm{dynamic}}$ are computed as reaction forces (opposite of elastic force) at interface nodes. At this stage, the displacement of all surface nodes of $\mathcal{M}_{quasi-static}$ is computed and the position of interface nodes becomes new displacement constraints for $\mathcal{M}_{\mathrm{dynamic}}$. After $\mathcal{M}_{quasi-static}$, $\mathcal{M}_{\mathrm{dynamic}}$ is updated based on displacements imposed at the interface nodes by $\mathcal{M}_{quasi-static}$ and the displacements imposed by the user interaction.

6.3.3 Examples

In Figure 35, we present an example of a hybrid cylinder model undergoing deformation caused by gravity forces. The different stages of the deformation process are shown. When the equilibrium is reached, as shown in the rightmost Figure, forces applied at the interface nodes are null and displacement vectors stabilize to a constant value. In this example, both quasi-static and dynamic models have the same elastic properties and we verified that the equilibrium position is the same as the one that would have been reached by a single quasi-static or dynamic elastic model. Furthermore, this hybrid model converges significantly faster than the corresponding dynamic elastic model.
The second example is related to the simulation of hepatectomy, *i.e* the removal of one of the eight anatomical segments – known as Couinaud segments [25] – of a liver. In this example the segment number six has to be removed. A tetrahedral

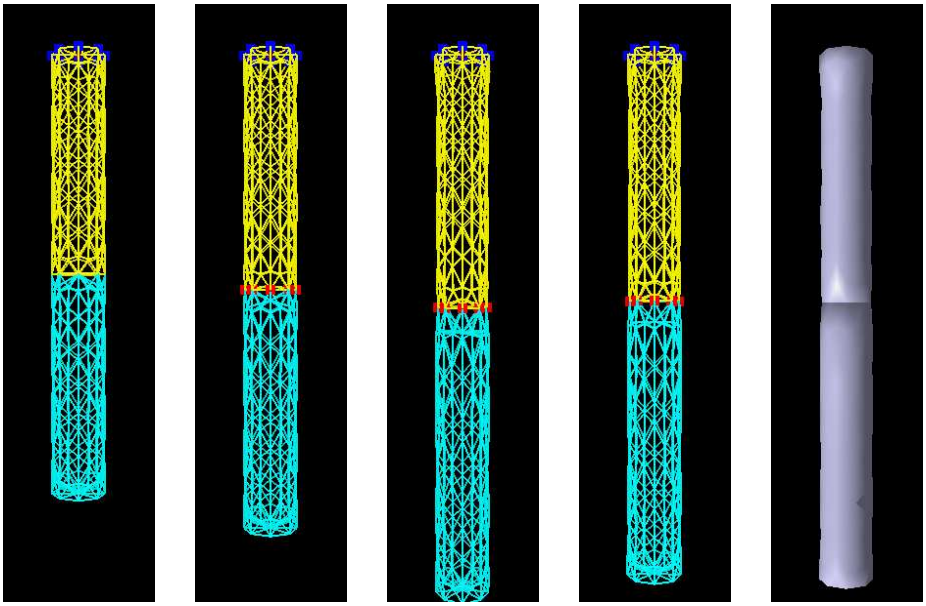


Figure 35: Deformation of a hybrid elastic model under a gravity force: the upper cylinder consists of a pre-computed linear elastic model whereas the lower part is a Tensor-Mass model. The leftmost figure corresponds to the initial position of the mesh and the rightmost figure to the equilibrium state.

mesh of a liver has been created from a CT scan image. It is composed of 1537 vertices and 7039 tetrahedra – see Figure 36. The tetrahedra of the sixth anatomical segment, which represent 18% (280 vertices and 1260 tetrahedra) of the global mesh, are modelled with a Tensor-Mass model and the remaining tetrahedra with a pre-computed linear elastic model.
In Figure 37, we show different stages of the hepatectomy simulation. The first six pictures show the deformation of the model when the tool collides with the dynamic model. Since both models have the same elastic characteristics, it is not possible to visually distinguish the interface between the two different elastic models.
The last six pictures show the cutting of the liver segment by removing additional tetrahedra. The cutting occurs for the tetrahedron being collided by the tool. One can notice that each part of the hybrid model deforms naturally itself during the resection simulation.

7 Large Displacement Non-Linear Elastic Model

7.1 Shortcomings of linear elasticity

The physical behavior of a soft tissue may be considered as linear elastic for small displacements and small deformations [41, 72]. The hypothesis of small displacements corresponds to displacements that are typically less than 10% of the mesh size.

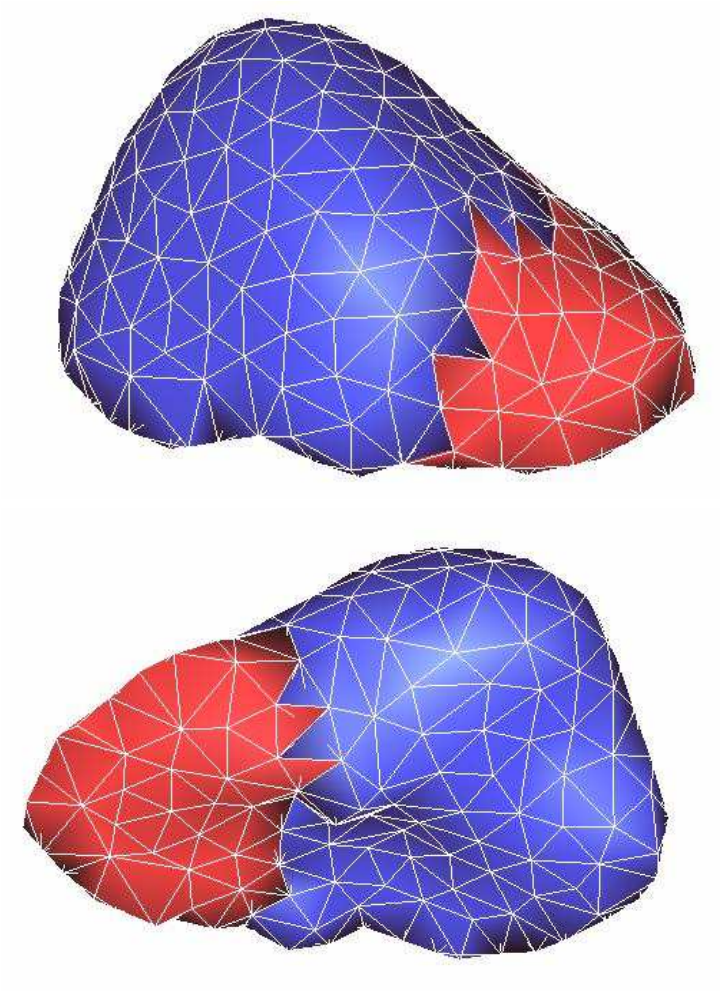


Figure 36: Display of a hybrid liver model. The part displayed in blue corresponds to the pre-computed quasi-static elastic model whereas the red part corresponds to the Tensor-Mass model. The interface nodes ensure the visual continuity between the two elastic models.

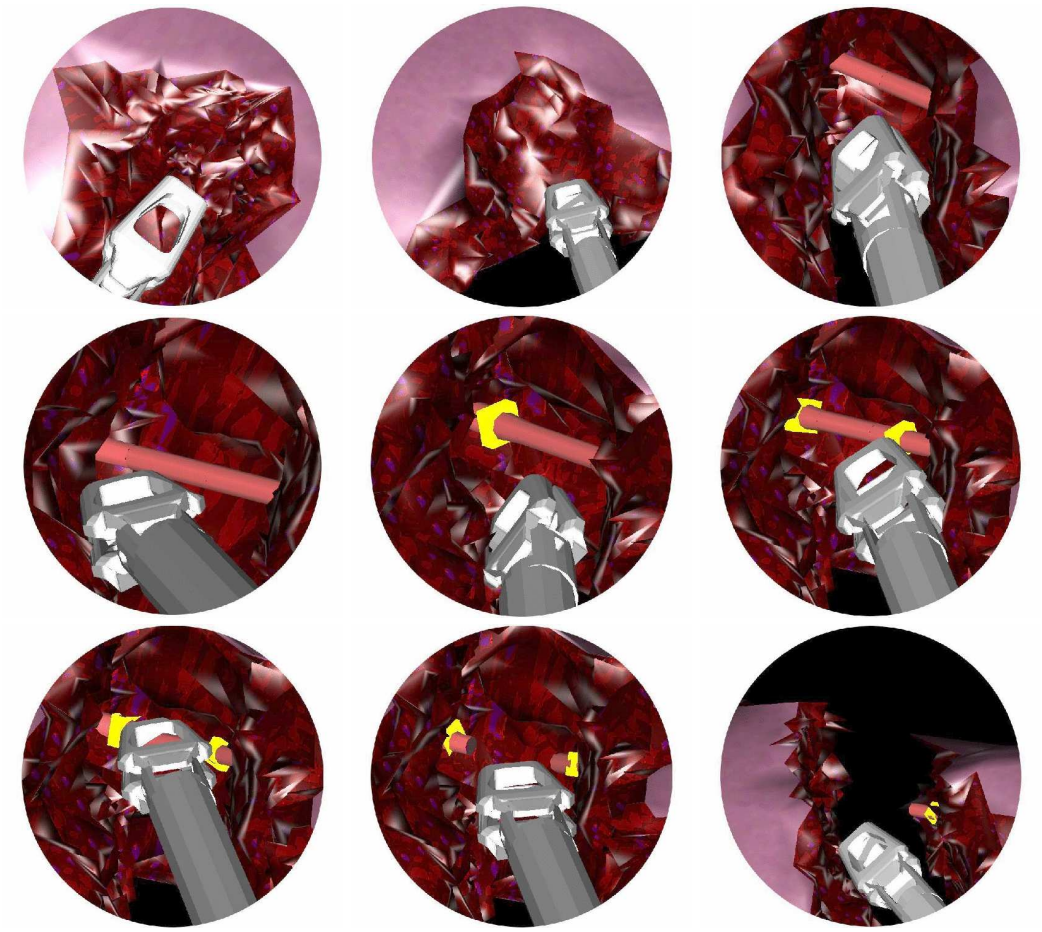


Figure 37: Different stages of the simulation of hepatectomy. In this simulation, we have included lineic models of the main bifurcations of the portal vein [38]. The simulation consists in removing some hepatic parenchyma but also to clamp and cut each vessel.

In the context of surgery simulation, this hypothesis is often violated. For instance, the lobes of the liver are often folded to access underlying structures such as the gall bladder. Also during the resection of a soft tissue, it is common that pieces being cut undergo large rotations either under the action of gravity or under the action of surgical instruments.
In such cases, linear elasticity is not an appropriate physical model because it makes the assumption of infinitesimal strain instead of finite strain. To exhibit the shortcomings of linear elasticity we produced two examples pictured in Figures 38 and 39.
In a first example, we illustrate the action of a global rotation on a linear elastic model. When an object (an icosahedron in Figure 38) undergoes a global rotation, its elastic energy increases, leading to a large variation of volume (as seen in the wireframe mesh of the rightmost figures). Indeed, the infinitesimal strain tensor $\mathbf{E}_{\mathrm{L}}(\mathbf{X}) = \frac{1}{2}(\nabla\mathbf{U} + \nabla\mathbf{U}^T)$ is not invariant when a global rotation $\mathbf{R}$ is applied since in this case $\nabla\mathbf{U} = \mathbf{R} - \mathbf{I}_3$ and therefore $\mathbf{E}_{\mathrm{L}}(\mathbf{X}) = \frac{1}{2}(\mathbf{R} + \mathbf{R}^T) - \mathbf{I}_3 \neq [\mathbf{0}]$. The two invariants $(tr\mathbf{E}_{\mathrm{L}})^2$ and $tr\mathbf{E}_{\mathrm{L}}^2$ increases under rotation as does the elastic energy.

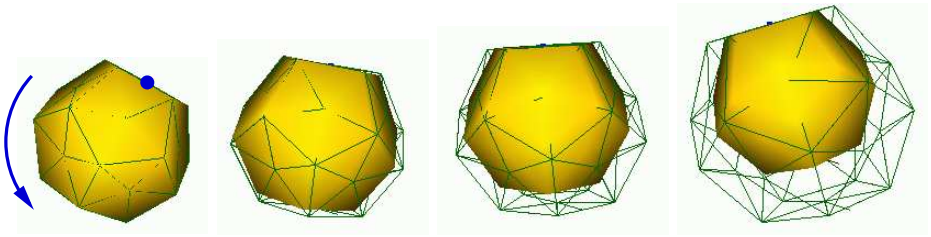
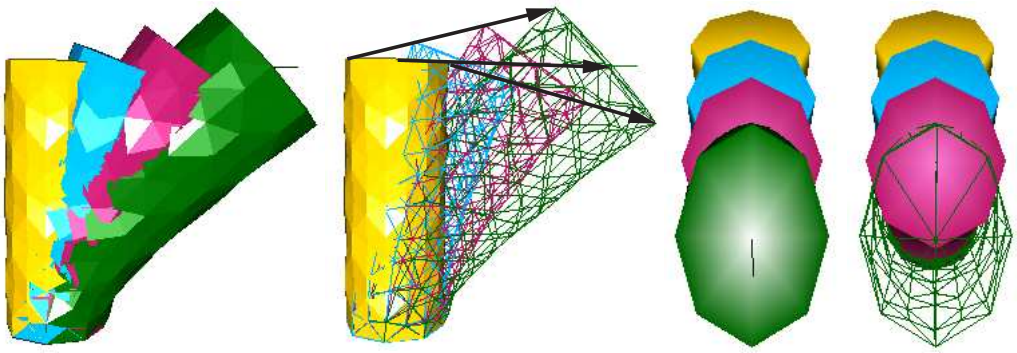


Figure 38: *Global rotation of the linear elastic model (wireframe)*



(a) (b) (c) (d)

Figure 39: *Successive deformations of a linear elastic cylinder [42]. (a) and (b): side view. (c) and (d): top view*

The second example shows the effect of linear elasticty when only one part of an object undergoes a large rotation (which is the most common case). The cylinder pictured in Figure 39 has its bottom face fixed while a force is being

applied at the central top vertex. The arrows correspond to the trajectories of some vertices : because of the linear elastic hypothesis, these trajectories are straight lines. This results in unrealistic distortions of the mesh. Moreover, abnormal deformations are not equivalent in all directions since the object only deforms itself in the rotation plane (figure 39(c) and 39(d)).

7.2 St Venant-Kirchhoff Elasticity

To overcome the limitations of linear elasticity, we proposed to adopt the St Venant-Kirchhoff elasticity. The St Venant-Kirchhoff model is a generalization of the linear model for large displacements, and is a particular case of hyperelastic materials. It has been used to model various materials (table 3.8.4 of Ciarlet [18] provides the constants for materials like steel, glass, lead or rubber) including facial soft tissue [44] and trabecular bone [7]. A St Venant Kirchhoff material relies on the Hooke's law as the definition of elastic energy (see equation 5 in section 3.2.3) but the linearized strain tensor $\mathbf{E}_{\mathrm{L}}$ is replaced by the Green-Lagrange strain tensor $\mathbf{E}$:

$$\mathbf{E}(\mathbf{X}) = \frac{1}{2}(\nabla\mathbf{U} + \nabla\mathbf{U}^T + \nabla\mathbf{U}^T\nabla\mathbf{U}) \tag{47}$$

$$W_{\mathrm{NL}}(\mathbf{X}) = \frac{\lambda}{2}\ (tr\mathbf{E})^2 + \mu\ tr\mathbf{E}^2 \tag{48}$$

The Green-Lagrange strain tensor $\mathbf{E}$ is no longer a linear function of the displacement field. A first property is that the elastic energy becomes invariant under the application of rotations. Indeed, when a rigid transformation (with rotation matrix $\mathbf{R}$) is applied to an object, the gradient of the displacement field is $\nabla\mathbf{U} = \mathbf{R} - \mathbf{I}_3$ and therefore the Green-Lagrange strain tensor remains zero (since $\mathbf{R}\,\mathbf{R}^T = \mathbf{I}_3$) :

$$\begin{aligned}
\mathbf{E}(\mathbf{X}) &= \frac{1}{2}\left(\mathbf{R} - \mathbf{I}_3 + \mathbf{R}^T - \mathbf{I}_3 + (\mathbf{R}^T - \mathbf{I}_3)(\mathbf{R} - \mathbf{I}_3)\right) \\
&= \frac{1}{2}\left(\mathbf{R} + \mathbf{R}^T - 2\mathbf{I}_3 + \mathbf{R}^T\mathbf{R} - \mathbf{R} - \mathbf{R}^T + \mathbf{I}_3\right) \\
&= [\mathbf{0}]
\end{aligned}$$

A second property is that the elastic energy W_{NL} (section 3.2.3), which was a quadratic function of $\nabla\mathbf{U}$ in the linear case, is now a fourth-order polynomial function with respect to $\mathbf{U}$:

$$\begin{aligned}
W_{\mathrm{NL}} &= \frac{\lambda}{2}\ (tr\mathbf{E})^2 + \mu\ tr\mathbf{E}^2 \qquad (49) \\
&= \frac{\lambda}{2}\left[(div\ \mathbf{U}) + \frac{1}{2}\|\nabla\mathbf{U}\|^2\right]^2 + \mu\ \|\nabla\mathbf{U}\|^2 - \frac{\mu}{2}\ \|rot\ \mathbf{U}\|^2 \\
&+ \mu(\nabla\mathbf{U} : \nabla\mathbf{U}^t\nabla\mathbf{U}) + \frac{\mu}{4}\|\nabla\mathbf{U}^t\nabla\mathbf{U}\|^2 \\
W_{\mathrm{NL}} &= W_{Linear} + \frac{\lambda}{2}\ (div\ \mathbf{U})\|\nabla\mathbf{U}\|^2 + \frac{\lambda}{8}\ \|\nabla\mathbf{U}\|^4 \\
&\qquad + \mu\ (\nabla\mathbf{U} : \nabla\mathbf{U}^t\nabla\mathbf{U}) + \frac{\mu}{4}\ \|\nabla\mathbf{U}^t\nabla\mathbf{U}\|^2,
\end{aligned}$$

where W_{Linear} is given by equation 5, and $A : B = tr(A^tB) = \sum_{i,j} a_{ij}\ b_{ij}$ is the dot product of two matrices.

Furthermore, we can extend this isotropic non-linear elastic energy to take into account "transversally isotropic" materials as performed in section 3.2.4 for the linear elastic model. In fact, equation 9, which defines the additional anisotropic term, still holds for St Venant-Kirchhoff elasticity. However, for the sake of clarity, we chose to keep only the anisotropic contribution which penalizes the material stretch in the direction given by unit vector $\mathbf{a}_0$:

$$W_{Trans\_iso} = W_{\mathrm{NL}} + \left(-\frac{\Delta\lambda}{2} + \Delta\mu\right)\left(\mathbf{a}_0{}^t\ \mathbf{E}\ \mathbf{a}_0\right)^2,$$

where $\Delta\lambda$ and Δ are the variations of Lamé coefficients along the direction of anisotropy.

7.3 Finite Element Modeling

By adopting the same methodology as the one presented in section 4.3, we provide a closed form expression of the elastic energy of a linear tetrahedron finite element :

$$\begin{aligned} W_{\mathrm{NL}}(\mathcal{T}) &= \frac{1}{2}\sum_{j,k}\mathbf{U}_j^t\left[\mathcal{B}_{jk}^{\mathcal{T}}\right]\mathbf{U}_k + \frac{1}{2}\sum_{j,k,l}\left(\mathbf{U}_j.\mathcal{C}_{jkl}^{\mathcal{T}}\right)\left(\mathbf{U}_k.\mathbf{U}_l\right) \\ &+ \frac{1}{2}\sum_{j,k,l,m}\mathcal{D}_{jklm}^{\mathcal{T}}\left(\mathbf{U}_j.\mathbf{U}_k\right)\left(\mathbf{U}_l.\mathbf{U}_m\right), \end{aligned} \tag{50}$$

where the terms $\mathcal{B}_{jk}^{\mathcal{T}}$, $\mathcal{C}_{jkl}^{\mathcal{T}}$, and $\mathcal{D}_{jklm}^{\mathcal{T}}$, called "stiffness parameters", are given by:

- $\mathcal{B}_{jk}^{\mathcal{T}}$ is a (3x3) symmetric matrix (which corresponds to the linear component of the energy):

$$\begin{aligned} 36V(\mathcal{T})\mathcal{B}_{jk}^{\mathcal{T}} &= \lambda\left(\mathbf{m}_j\otimes\mathbf{m}_k\right) + \mu\left[\left(\mathbf{m}_k\otimes\mathbf{m}_j\right) + \left(\mathbf{m}_j.\mathbf{m}_k\right)\mathbf{I}_3\right] \\ &+ \left(-\frac{\Delta\lambda}{2} + \Delta\mu\right)(\mathbf{a}_0\otimes\mathbf{a}_0)(\mathbf{m}_j\otimes\mathbf{m}_k)(\mathbf{a}_0\otimes\mathbf{a}_0), \end{aligned}$$

- $\mathcal{C}_{jkl}^{\mathcal{T}}$ is a vector:

$$\begin{aligned} 216(V(\mathcal{T}))^2\mathcal{C}_{jkl}^{\mathcal{T}} &= \frac{\lambda}{2}\mathbf{m}_j\left(\mathbf{m}_k.\mathbf{m}_l\right) + \frac{\mu}{2}\left[\mathbf{m}_l\left(\mathbf{m}_j.\mathbf{m}_k\right) + \mathbf{m}_k\left(\mathbf{m}_j.\mathbf{m}_l\right)\right] \\ &+ \left(-\frac{\Delta\lambda}{2} + \Delta\mu\right)(\mathbf{a}_0\otimes\mathbf{a}_0)(\mathbf{m}_j\otimes\mathbf{m}_k)(\mathbf{a}_0\otimes\mathbf{a}_0)\mathbf{m}_l, \end{aligned}$$

- and $\mathcal{D}_{jklm}^{\mathcal{T}}$ is a scalar:

$$\begin{aligned} 1296(V(\mathcal{T}))^3\mathcal{D}_{jklm}^{\mathcal{T}} &= \frac{\lambda}{8}\left(\mathbf{m}_j.\mathbf{m}_k\right)\left(\mathbf{m}_l.\mathbf{m}_m\right) + \frac{\mu}{4}(\mathbf{m}_j.\mathbf{m}_m)(\mathbf{m}_k.\mathbf{m}_l) \\ &+ \frac{1}{4}\left(-\frac{\Delta\lambda}{2} + \Delta\mu\right)(\mathbf{a}_0.\mathbf{m}_j)(\mathbf{a}_0.\mathbf{m}_k)(\mathbf{a}_0.\mathbf{m}_l)(\mathbf{a}_0.\mathbf{m}_m). \end{aligned}$$

- The last term of each stiffness parameter models the anisotropic behavior of the material.

The elastic force applied at each vertex $\mathbf{p}_i$ of tetrahedron $\mathcal{T}$ is obtained as the derivation of the elastic energy $W_{\mathrm{NL}}(\mathcal{T})$ with respect to the displacement $\mathbf{p}_i$:

$$\begin{aligned}\mathbf{F}_i(\mathcal{T}) &= \underbrace{\sum_j \left[\mathcal{B}_{ij}^{\mathcal{T}}\right] \mathbf{U}_j}_{\mathbf{F}_i^1(\mathcal{T})} + \underbrace{\sum_{j,k} \left(\mathbf{U}_k \otimes \mathbf{U}_j\right) \mathcal{C}_{jki}^{\mathcal{T}} + \frac{1}{2} \left(\mathbf{U}_j.\mathbf{U}_k\right) \mathcal{C}_{ijk}^{\mathcal{T}}}_{\mathbf{F}_i^2(\mathcal{T})} \\ &+ \underbrace{2 \sum_{j,k,l} \mathcal{D}_{jkli}^{\mathcal{T}} \mathbf{U}_l \mathbf{U}_k^t \mathbf{U}_j}_{\mathbf{F}_i^3(\mathcal{T})}.\end{aligned} \tag{51}$$

The first term of the elastic force ($\mathbf{F}_i^1(\mathcal{T})$) corresponds to the linear elastic case presented in section 4.4.

7.4 Non-linear Tensor-Mass Model

In this section, we generalize the Tensor-Mass model introduced in 6.1 to the case of large displacement elasticity. The only changes in the Tensor-Mass algorithm are related to the computation of the elastic force $\mathbf{F}_i$ applied at vertex i.

In the case of linear elasticity, this force was computed by a first scan of all edges to compute the terms $[\mathbf{K}_{ij}]\mathbf{u}_j$ followed by a scan of all vertices to add the terms $[\mathbf{K}_{ii}]\mathbf{u}_i$.

We proposed to apply the same principle to the quadratic term ($\mathbf{F}_2^p(\mathcal{T})$ of equation 51) and the cubic term ($\mathbf{F}_3^p(\mathcal{T})$). The former requires **stiffness vectors** for vertices, edges, and triangles, and the latter requires **stiffness scalars** for vertices, edges, triangles, and tetrahedra.

The task of assembling global stiffness parameters is slightly more time consuming than in the linear case, since 31 parameters must be assembled instead of 2 ; these parameters are presented in table 6.

For vertex, edge and triangle parameters, one needs to add the contributions of all neighboring tetrahedra. For instance, the vertex rigidity vector $\mathcal{C}^{ppp}$ is computed at vertex p as :

$$\mathcal{C}^{ppp} = \sum_{\mathcal{T} \in \mathcal{S}(p)} \mathcal{C}_{ppp}^{\mathcal{T}}$$

For the 6 scalar parameters $\mathcal{D}^{jklp}$ stored at each tetrahedron, no assembly is required since there is no other contribution originating from another tetrahedron.

The computation of the elastic force is performed by successively scanning tetrahedra, triangles, edges and vertices of the mesh. When scanning triangles for instance, the contributions from the three triangles are computed and added to the elastic force of each of its three vertices. The contribution for each element is summarized in equation 51.

$$\mathbf{F}_i = \mathbf{F}_i^{vertex} + \mathbf{F}_i^{edge} + \mathbf{F}_i^{triangle} + \mathbf{F}_i^{tetrahedron} \tag{52}$$

Stiffness parameters distribution	Tensors	Vectors	Scalars
Vertex p	$\mathcal{B}^{pp}$	$\mathcal{C}^{ppp}$	$\mathcal{D}^{pppp}$
Edge (p,j)	$\mathcal{B}^{pj}$	$\mathcal{C}^{ppj}$ $\mathcal{C}^{jpp}$ $\mathcal{C}^{jjp}$ $\mathcal{C}^{pjj}$	$\mathcal{D}^{jppp}$ $\mathcal{D}^{jjjp}$ $\mathcal{D}^{jpjp}$ $\mathcal{D}^{pjjp}$ $\mathcal{D}^{jjpp}$
Triangle (p,j,k)		$\mathcal{C}^{jkp}$ $\mathcal{C}^{kjp}$ $\mathcal{C}^{pjk}$	$\mathcal{D}^{jkpp}$ $\mathcal{D}^{jpkp}$ $\mathcal{D}^{pjkp}$ $\mathcal{D}^{jjkp}$ $\mathcal{D}^{jkjp}$ $\mathcal{D}^{kjjp}$ $\mathcal{D}^{kkjp}$ $\mathcal{D}^{kjkp}$ $\mathcal{D}^{jkkp}$
Tetrahedron (p,j,k,l)			$\mathcal{D}^{jklp}$ $\mathcal{D}^{jlkp}$ $\mathcal{D}^{kjlp}$ $\mathcal{D}^{kljp}$ $\mathcal{D}^{ljkp}$ $\mathcal{D}^{lkjp}$

Table 6: *Storage of the stiffness parameters on the mesh*

with :

$$\mathbf{F}_i^{vertex} = \boxed{\begin{array}{l} \text{Vertex contribution} \\ \\ \quad [\mathcal{B}^{pp}]\,\mathbf{U}_p \\ + \quad \left[(\mathbf{U}_p \otimes \mathbf{U}_p) + \frac{1}{2}\,(\mathbf{U}_p.\mathbf{U}_p)\,\mathbf{I}_3\right]\mathcal{C}^{ppp} \\ + \quad 2\mathcal{D}^{pppp}\,\mathbf{U}_p\,\mathbf{U}_p^t\,\mathbf{U}_p \end{array}}$$

$$\mathbf{F}_i^{edge} = \sum_{edges(p,j)} \boxed{\begin{array}{l} \text{Edge contribution} \\ \\ \quad \left[\mathcal{B}^{pj}\right]\mathbf{U}_j \\ + \quad [(\mathbf{U}_j \otimes \mathbf{U}_p) + (\mathbf{U}_j.\mathbf{U}_p)\,\mathbf{I}_3]\,\mathcal{C}^{ppj} + (\mathbf{U}_p \otimes \mathbf{U}_j)\,\mathcal{C}^{jpp} \\ \quad + (\mathbf{U}_j \otimes \mathbf{U}_j)\,\mathcal{C}^{jjp} + \frac{1}{2}\,(\mathbf{U}_j.\mathbf{U}_j)\,\mathcal{C}^{pjj} \\ + \quad 2\left[\mathcal{D}^{jppp}\,\left(2\mathbf{U}_p\,\mathbf{U}_p^t\,\mathbf{U}_j + \mathbf{U}_j\,\mathbf{U}_p^t\,\mathbf{U}_p\right) + \mathcal{D}^{jjpp}\,\mathbf{U}_p\,\mathbf{U}_j^t\,\mathbf{U}_j\right. \\ \quad \left. + \left(\mathcal{D}^{jpjp} + \mathcal{D}^{pjjp}\right)\mathbf{U}_j\,\mathbf{U}_j^t\,\mathbf{U}_p + \mathcal{D}^{jjjp}\,\mathbf{U}_j\,\mathbf{U}_j^t\,\mathbf{U}_j\right] \end{array}}$$

$$\mathbf{F}_i^{triangle} = \sum_{faces(p,j,k)}$$

Triangle contribution
$\big[(\mathbf{U}_k \otimes \mathbf{U}_j)\,\mathcal{C}^{jkp} + (\mathbf{U}_j \otimes \mathbf{U}_k)\,\mathcal{C}^{kjp} + (\mathbf{U}_j.\mathbf{U}_k)\,\mathcal{C}^{pjk}\big]$ $+2\left[\left(\mathcal{D}^{pjkp} + \mathcal{D}^{jpkp}\right)\left(\mathbf{U}_j\ \mathbf{U}_k^t\ \mathbf{U}_p + \mathbf{U}_k\ \mathbf{U}_j^t\ \mathbf{U}_p\right) + 2\mathcal{D}^{jkpp}\ \mathbf{U}_p\ \mathbf{U}_j^t\ \mathbf{U}_k\right.$ $+\left(\mathcal{D}^{kjjp} + \mathcal{D}^{jkjp}\right)\mathbf{U}_j\ \mathbf{U}_j^t\ \mathbf{U}_k + \mathcal{D}^{jjkp}\ \mathbf{U}_k\ \mathbf{U}_j^t\ \mathbf{U}_j$ $+\left(\mathcal{D}^{jkkp} + \mathcal{D}^{kjkp}\right)\mathbf{U}_k\ \mathbf{U}_k^t\ \mathbf{U}_j + \mathcal{D}^{kkjp}\ \mathbf{U}_j\ \mathbf{U}_k^t\ \mathbf{U}_k\big]$

$$\mathbf{F}_i^{tetrahedron} = \sum_{tetra(p,j,k,l)}$$

Tetrahedron contribution
$2\left[\left(\mathcal{D}^{jklp} + \mathcal{D}^{kjlp}\right)\mathbf{U}_l\ \mathbf{U}_j^t\ \mathbf{U}_k + \left(\mathcal{D}^{jlkp} + \mathcal{D}^{ljkp}\right)\mathbf{U}_k\ \mathbf{U}_j^t\ \mathbf{U}_l\right.$ $\left.+\left(\mathcal{D}^{kljp} + \mathcal{D}^{lkjp}\right)\mathbf{U}_j\ \mathbf{U}_k^t\ \mathbf{U}_l\right]$

In terms of data structure, the non-linear Tensor-Mass model requires the addition of triangles in the mesh topological description. In our case, we chose to store triangles in a hash table which is hashed by the three indices of its vertices in lexicographic order. Furthermore, each tetrahedron owns pointers towards its four triangles and reversely, each triangle owns pointers towards its two neighboring tetrahedra.
During the simulation of resection, tetrahedra are iteratively removed near the extremities of virtual cavitron instruments. When removing a single tetrahedron, 280 floating point numbers are updated to suppress the tetrahedron contributions to the stiffness parameters of the surrounding vertices, edges, and triangles :

$$\begin{aligned} & 4 * (1 \text{ tensor} + 1 \text{ vector} + 1 \text{ scalar}) \\ + \quad & 6 * (1 \text{ tensor} + 4 \text{ vectors} + 5 \text{ scalars}) \\ + \quad & 4 * (3 \text{ vectors} + 9 \text{ scalars}) \\ = \quad & 280 \text{ real numbers} \end{aligned}$$

By locally updating stiffness parameters, the tissue has exactly the same properties as if the corresponding tetrahedron had been removed at its rest position. Because of the volumetric continuity of finite element modeling, the tissue deformation remains realistic during cutting.

7.5 Incompressibility constraint

Living tissue, which is made essentially of water is almost incompressible, a property which is difficult to model and which, in most cases, leads to instability problems. This is the case with the St Venant-Kirchhoff model: the material remains incompressible when the Lamé constant λ tends towards infinity. Taking a large value for λ would impose to decrease the time step and therefore to increase the computation time. Another reason to add an external incompressibility constraint to the model is intrinsic to the model itself : the St Venant-Kirchhoff model relies on the Green-Lagrange strain tensor E which is invariant with respect to rotations. But it is also invariant with respect to symmetries, which could lead to the reversal of some tetrahedra under strong constraints.

We chose to penalize volume variation by applying to each vertex of the tetrahedron a force directed along the normal of the opposite face $\mathbf{N}_p$ (see figure 40), the norm of the force being proportional to the square of the relative volume variation:

$$\mathbf{F}^p_{incomp} = sign(V - V_0) \left(\frac{V - V_0}{V_0} \right)^2 \vec{\mathbf{N}}_p. \tag{53}$$

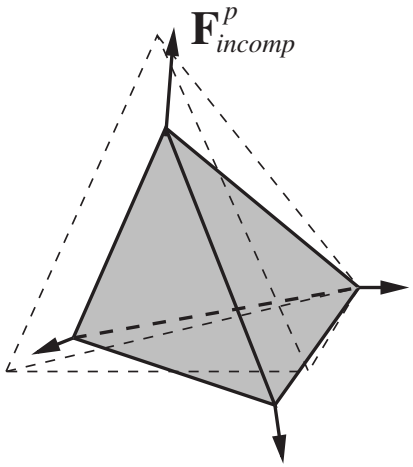
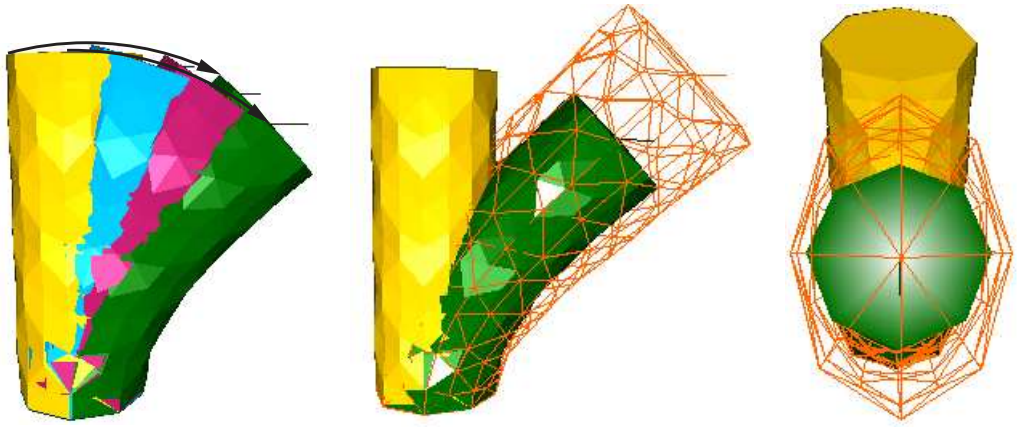


Figure 40: *Penalization of volume variation*

Since the volume V is proportional to the height of each vertex facing its opposite triangle, when V is greater than V_0 then the force $\mathbf{F}^p_{incomp}$ tends to decrease V by moving each vertex along the normal of the triangle triangle facing it. These forces act as an artificial pressure inside each tetrahedron. This method is closely related to Lagrange multipliers, which are often used to solve problem of energy minimization under constraints.

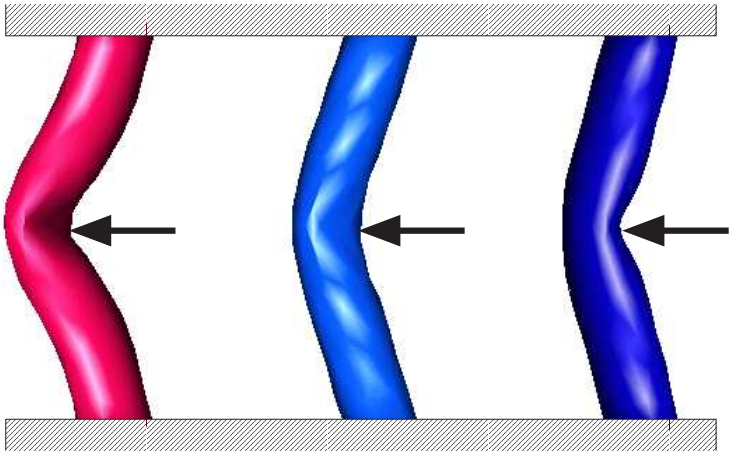
7.6 Results

In a first experiment, we wish to highlight the contributions of our new deformable model in the case of partial rotations. Figure 41 shows the same experience as the one presented for linear elasticity (section 7.1, Figure 39). On the left we can see that the cylinder vertices are now able to follow non-straight trajectories (figure 41(a)), leading to much more realistic deformations than in the linear (wireframe) case (figures 41(b) and 41(c)).
The second example presents the differences between isotropic and anisotropic materials. The three cylinders of figure 42 have their top and bottom faces fixed, and are submitted to the same forces. While the isotropic model on the left undergoes a "snake-like" deformation, the last two, which are anisotropic along their height, stiffen in order to minimize their stretch in the anisotropic direction. The rightmost model, being twice as stiff as the middle one in the anisotropic direction, starts to squeeze in the plane of isotropy because it cannot stretch anymore.



(a) (b) (c)

Figure 41: *(a) Successive deformations of the non-linear model [82]. Side (b) and top (c) view of the comparison between linear (wireframe) and non-linear model (solid rendering)*



$\lambda = 2.10^5\ kg/cm^2$	$\lambda = 2.10^5\ kg/cm^2$	$\lambda = 2.10^5\ kg/cm^2$
$\mu = 10^5\ kg/cm^2$	$\mu = 10^5\ kg/cm^2$	$\mu = 10^5\ kg/cm^2$
-	$\lambda^L = 10^6\ kg/cm^2$	$\lambda^L = 2.10^6\ kg/cm^2$
-	$\mu^L = 5.10^5\ kg/cm^2$	$\mu^L = 10^6\ kg/cm^2$

Figure 42: Shearing deformation of tubular structures under the action of the force indicated by the arrow. The leftmost figure corresponds to an isotropic non-linear material while the center and rightmost figures correspond to a non-linear anisotropic material, the direction of anisotropy being the cylinder axis.

In the third example (figure 43), we apply a force to the right lobe of the liver (the liver is fixed in a region near the center of its back side, and Lamé coefficients are: $\lambda = 40kPa$ and $\mu = 10kPa$). Using the linear elastic model, the right part of the liver undergoes a large (and unrealistic) volume increase, whereas with non-linear elasticity, the right lobe is able to rotate partially, while adopting a more realistic deformation.

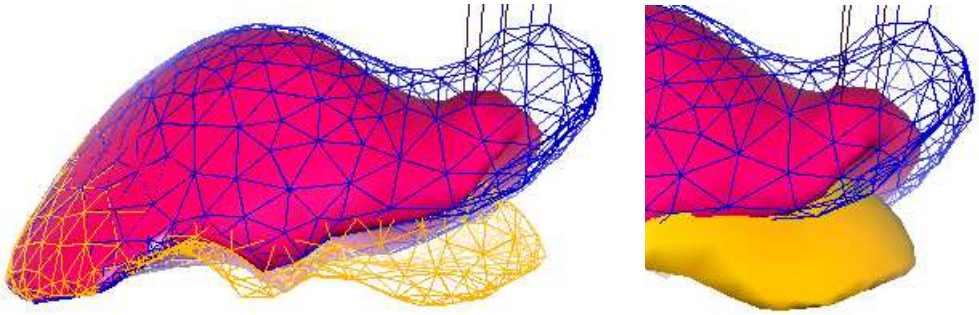


Figure 43: *Linear (upper mesh in wireframe), non-linear (Gauraud shaded) liver models, and rest shape (lower mesh in wireframe). In both cases, the same forces showed in solid lines are applied to three surface nodes lying on the left lobe [82].*

Adding the incompressibility constraint on the same examples decreases the volume variation even more (see table 7), and also stabilizes the behaviour of the deformable models in highly constrained areas.

Volume variation (%)	Linear	Non-linear	Non-linear incomp.
Cylinder **left** | **middle** | **right**	7 | 28 | 63	0.3 | 1 | 2	0.2 | 0.5 | 1
Liver	9	1.5	0.7

Table 7: *Volume variation results. For the cylinder: left, middle and right stand for the different deformations of Figures 42 and 43.*

The last example is the simulation of a typical laparoscopic surgical gesture on the liver. One tool is pulling the edge of the liver sideways while a bipolar cautery device cuts it. During the cutting, the surgeon pulls away the part of the liver he wants to remove. This piece of liver undergoes large displacements and the deformation appears fairly realistic with this new non-linear deformable model (figure 44).
Obviously, the computation time of this model is larger than for the linear model because the force equation is more complex (equation 51 in section 7.3 to be

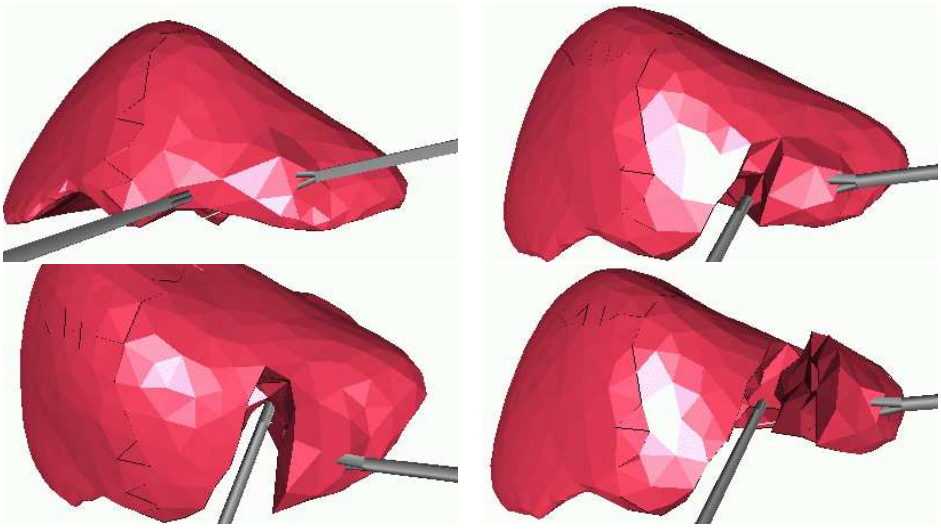


Figure 44: *Simulation of laparoscopic liver surgery*

compared with equation 41 in section 6.1.7). With our current implementation, the simulation refresh rate is five times slower than with the linear model. Nevertheless, with this non-linear model, we can reach an update cycle of 25 Hz on meshes made of about 2000 tetrahedra (on a PC Pentium PIII 500M Hz). This is enough to achieve real-time visual feedback with quite complex objects, and even to provide a realistic haptic feedback using force extrapolation as described in [83].

7.7 Optimization of non-linear deformations

We showed that non-linear elasticity allows to simulate much more realistic deformations than linear elasticity when the model undergoes large displacements. However, non-linear elasticity is more computationally expensive than linear elasticity. Since non-linear elastic forces tend to linear elastic forces as the maximum vertex displacement decreases to zero, we propose to use non-linear elasticity only at parts of the mesh where displacements are larger than a given threshold, the remaining part using linear elasticity. Thus, we modified the force computation algorithm in the following manner : for each vertex, we first compute the linear part of the force, and we add the non-linear part only if its displacement is larger than a threshold. Figure 45 shows a deformation computed with this optimization (same model as in figure 43). This liver model is made of 6342 tetrahedra and 1394 vertices. The threshold is set to 2 cm while the mesh is about 30 cm long. The points drawn on the surface identify vertices using non-linear elasticity. With this method, we reach an update frequency of 20 Hz instead of 8 Hz for a fully non-linear model. The same deformation is presented on figure 46 for different values of the threshold. With this method, we can choose a trade-off between the bio-mechanical realism of the deformation and the update frequency of the simulation. The diagram on figure 47 shows the update frequencies reached for each value of the threshold, in comparison with

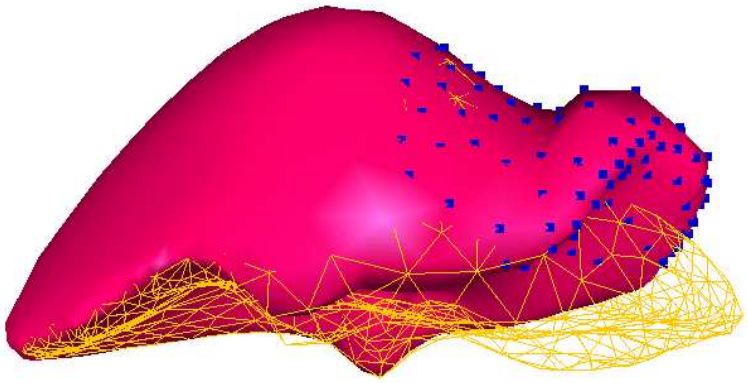
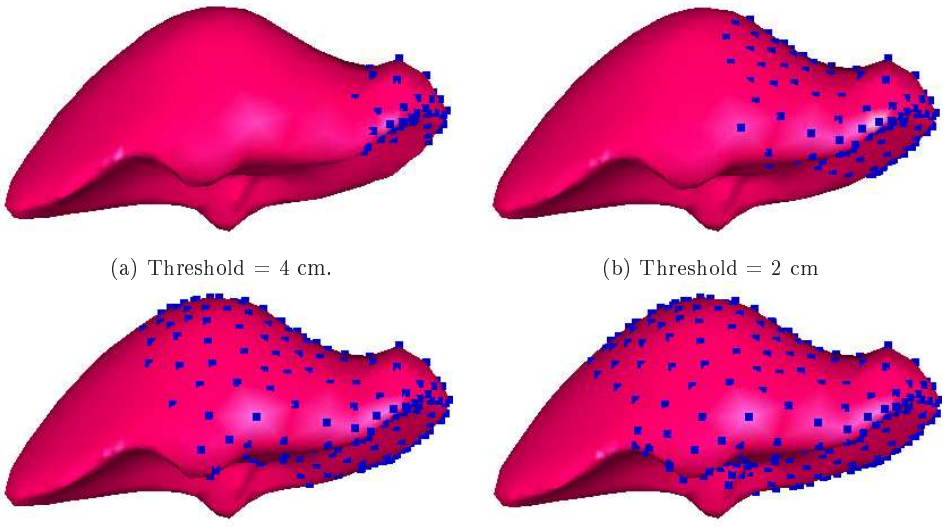


Figure 45: *Adaptable non-linear model deformation compared to its rest position (wireframe)*



(a) Threshold = 4 cm.

(b) Threshold = 2 cm

(c) Threshold = 1 cm

(d) Threshold = 0.5 cm

Figure 46: *Deformation of the adaptive non-linear model for several values of the threshold.*

the fully linear and the fully non-linear models. Even when this threshold tends towards infinity, the adaptable model is slower than the linear model, because the computation algorithm of the non-linear force is more complex. Indeed, the computation of non-linear forces requires to visit all vertices, edges, triangles, and tetrahedra of the mesh, whereas only vertices and edges need to be visited for the linear model. For the simulation example of figure 44, this optimization

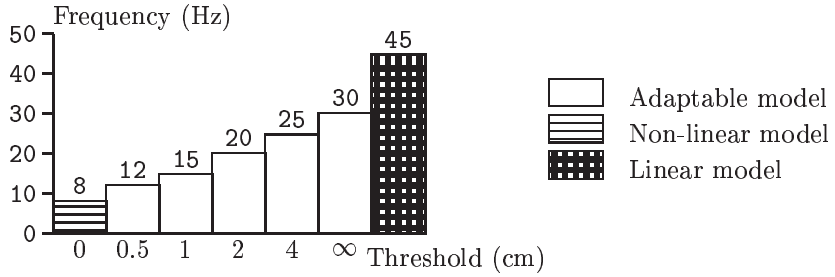


Figure 47: *Updating frequencies of the adaptable model for several values of the threshold*

leads to update frequencies varying between 50 and 80 Hz, depending on the number of points modeling non-linear elasticity (figure 48). The minimal frequency of 50 Hz is reached at the end of the simulation, when all vertices of the resected part of the liver are using large displacement elasticity (on the right of figure 48).
In general, two strategies can be used to set the value of this threshold. In the first strategy, the threshold is increased until a given update frequency is matched as demonstrated previously. The second strategy is physically-motivated and sets the threshold to 10 % of the typical size of the mesh since it corresponds to the extent of displacement for which linear elasticity remains a valid constitutive law.

8 Conclusion

In this chapter, we have presented several algorithms for computing in real-time the deformation of soft tissues in a surgical simulator. We wish to stress two important aspects of these algorithms. First of all, using linear tetrahedra as finite elements helped us to write closed-form expressions of the elastic energy and its derivatives, even in the case of large displacement elasticity. These expressions nicely decouple the physical parameters (Lamé coefficients) from the geometry of each tetrahedron both in its rest position (direction of anisotropy, rest volume, area vectors) and in its deformed state (displacement vectors). Furthermore, it enables to quickly assemble local and global stiffness matrices when the mesh topology has been modified during a cutting simulation.
Second, in the context of surgery simulation, soft tissue deformation algorithms are closely tied with the visualization, collision detection and haptic rendering algorithms. Furthermore, the traditional stages of matrix assembly, matrix preconditioning, system solution and post-processing, cannot be easily decoupled like in classical software packages available in structural mechanics. This

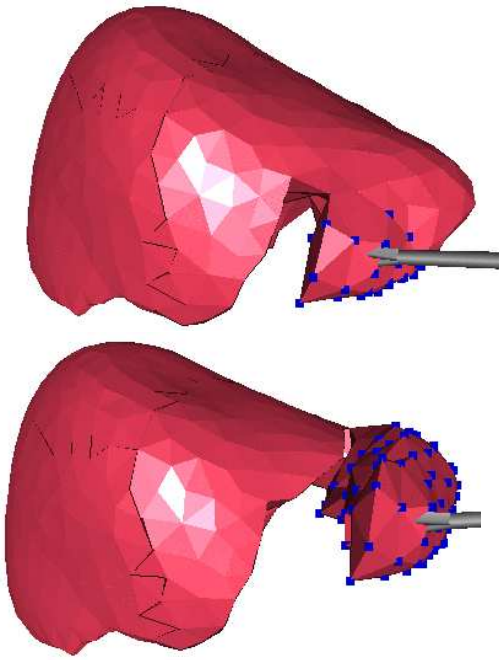


Figure 48: *Simulation of hepatectomy based on a non-linear adaptable elastic model. non-linear elastic force are applied on vertices outlined with a box.*

implies that the data structure and the flow chart must be carefully designed in order to achieve a reasonable trade-off between these performances. Therefore, building a successful simulator can only be achieved by a multidisciplinary effort covering the fields of biomechanics, numerical analysis, robotics and computer graphics.

An hepactectomy simulator based on the quasi-static precomputed linear elastic model (introduced in section 5) and the large displacement non-linear elastic models (introduced in section 7) has been built where the following three basic surgical gestures can be rehearsed : touching soft tissue, gripping soft tissue, and cutting parenchyma with a cavitron. Furthermore we recently added a physical model of the portal vein [36], which allows the user to simulate the clamping and cutting of vessels during the hepatic resection.

However, to increase the training impact and realism of the simulation, it is important to simulate the contact between the liver and neighboring structures such as the gall-bladder, the different ligaments, the right kidney, the peritoneum, etc. These additional surface and volumetric models require to extend the soft tissue models introduced in this chapter in two ways.

First, it is necessary to extend the precomputed linear elastic model to include large-displacement non-linear elasticity. Indeed, the linear domain of biological soft tissue is usually rather small, and therefore many surgical gestures can only be simulated by using large-displacement elasticity (like rotating the lobe of the liver or resecting the gall-bladder). The precomputation of non-linear elastic material is not a trivial task since it implies solving a complex third-order al-

gebraic equation in the case of St Venant-Kirchhoff elasticity (see section 7.2). Instead, it may be possible to find suitable approximations which can be computed efficiently.
Second, it is necessary to extend the concept of hybrid models (introduced in section 6.3) in order to cope with the deformation of models including several tens of thousands of vertices. Ideally, we would like to provide accurate but computationnally expensive soft tissue models in the center of the surgical field where the user performs complex gestures and at the same time to provide less expensive models but potentially less accurate, away from the center of the surgical field. Of course during surgery, the focus of the surgeon may switch from the gall-bladder to the hepatic parenchyma which implies that those tissue models should evolve dynamically from one level of accuracy to the other. Achieving this level of scalability with the constraint that the topology of these models may change over time, is the main challenge of soft tissue modeling for surgery simulation.
Finally, we would like to stress the importance of validating the different components of a surgical simulator. Concerning soft tissue models, there are at least three levels of validation that need to be achieved. A first validation consists in comparing the soft tissue deformation algorithms that rely on strong hypotheses against well-known finite element packages in order to evaluate the range of approximations that are performed. In the second level of validation, the biomechanical behavior of each anatomical structure must be compared to experimental dataset. Ideally, one would like to validate both boundary conditions and the constitutive law of each biological tissue. However, in practice, this validation is made difficult by the lack of quantitative experimental information. The third level of validation consists in evaluating the dynamic behaviour of each soft tissue during the simulation since some models that appear too soft or too stiff. Finally, and most importantly, it is required to validate the whole simulation system by assessing its ability to succeed in training young residents to perform a given surgical task.
Despite these remaining issues to be solved, we believe that practical surgery simulators will be fully operational and actually part of the surgical studies in the near future.

Acknowledgments

We would like to thank Matthias Teschner, Denis Laurendeau and Jean-Marc Schwartz for their priceless comments and for proofreading this article.
The work presented in this paper is a joint work between the authors and mainly two former PhD students : Stéphane Cotin and Guillaume Picinbonno. Stéphane Cotin developped the precomputed linear elastic model of section 5 as well as a first version of the Tensor-Mass model described in section 6.1. Guillaume Picinbonno proposed the extension of the Tensor-Mass model to the case of large displacement elasticity (in section 7). We also wish to thank Clément Forest and Jean-Christophe Lombardo for their numerous contributions on force-feedback rendering, collision detection as well as mesh data structure. This work was fueled with the stimulating remarks and propositions from our INRIA colleagues who participated in the AISIM and CAESARE joint initiatives : Marie-Paule Cani, Marina Vidrascu, Marc Thiriet, Christian Laugier.

Also, we are grateful to Pr. Marescaux, Pr. Leroy, and Pr. Luc Soler from the IRCAD research center for their long-term vision and for sharing their expertise of abdominal surgery with us. Finally, we would like to acknowledge the strong support we received from Gilles Khan, INRIA Vice-President for Research, during the different stages of this research work.

Keyword Index

List of Mathematical Symbols

f_u	Update frequency of the soft tissue model
t	Discrete or continuous time variable
X_t	Position the model at time t
$T_{relaxation}$	Relaxation time
T_c	Computation time
$T_{interaction}$	Latency caused by the software and harware architecture
Δt	Time step used in the discretization of temporal derivatives
$\mathbf{F}$	Global force vector
$\mathbf{K}$	Global stiffness matrix
$\mathbf{U}$	Global displacement vector
$\mathbf{M}$	Global mass matrix
$\mathbf{C}$	Global damping matrix
$\dot{\mathbf{U}}$	Global speed vector
$\mathcal{M}_{rest}$	Soft tissue Model at its rest position
$\mathcal{M}_{def}$	Soft tissue Model at its deformed position
Ω	Region of space for the rest configuration
$\mathbf{\Phi}(x, y, z)$	Deformation function that maps point (x, y, z) from the rest configuration to the deformed configuration
$\mathbf{X}$	Point in the rest configuration
$\mathbf{U}(\mathbf{X})$	Displacement function
$\mathbf{C}(\mathbf{X})$	Right Cauchy-Green strain tensor
$\mathbf{E}(\mathbf{X})$	Green-Lagrange strain tensor
$\mathbf{I}_3$	3×3 identity matrix
$\mathbf{E}_{\mathrm{L}}$	Linearized strain tensor
e_{ij}	Element of the linearized strain tensor
$\mathbf{T}(\mathbf{X})$	Cauchy stress tensor
$W(\mathbf{X})$	Density of elastic energy
λ, μ	Isotropic Lamé coefficients
E, ν	Isotropic Young modulus and Poisson ratio
$\mathbf{a}_0$	Unit vector along the direction of anisotropy for transverally isotropic materials
$\lambda^{\mathbf{a}_0}, \mu^{\mathbf{a}_0}$	Lamé coefficients along the direction of anisotropy
$\Delta\lambda, \Delta\mu$	Difference between the Lamé coefficients along the direction of anisotropy and those in the orthogonal plane
$\Delta W_{Ani}(\mathbf{X})$	Additional term of the density of elastic energy caused by anisotropy
I_4, I_5	Deformation invariants estimated along the direction of anisotropy
$\mathbf{p}_i$	Point of a tetrahedron in its rest position
$\mathbf{q}_i$	Point of a tetrahedron in its deformed position
$\mathbf{u}_i$	Displacement vector of a vertex of a tetrahedron
$\mathcal{T}$	Tetrahedron as a linear finite element
$h_j(\mathbf{X})$	Shape functions associated with a linear tetrahedron

$\mathbf{P}$	4×4 matrix describing the shape functions
$V(\mathcal{T})$	Volume of tetrahedron $\mathcal{T}$
$\mathbf{m}_i$	Area vector opposite to vertex i
V_i	6 times the volume of the tetrahedron made by the origin $\mathbf{o}$ and vertices $\mathbf{p}_{i+1}$, $\mathbf{p}_{i+2}$ and $\mathbf{p}_{i+3}$
T_i	Triangle opposite to vertex i
$\mathbf{n}_i$	Normal vector at the triangle T_i opposite to vertex i in a tetrahedron
$\theta_{i,j}$	Angle between normal vectors of triangles T_i and T_j
A_i	Area of triangle T_i
$l_{i,j}$	Length of the edge connecting vertices i and j
f_i	Height of vertex o above triangle T_i
$\mathbf{B}^{\mathcal{T}}_{i,j}$	Element (i,j) of the 3×3 stiffness matrix for a tetrahedron $\mathcal{T}$ made of an isotropic material
$\mathbf{A}^{\mathcal{T}}_{i,j}$	Element (i,j) of the 3×3 stiffness matrix for a tetrahedron $\mathcal{T}$ made of an transversally isotropic material
$\mathbf{K}_{i,j}$	3×3 global stiffness matrix between vertex i and j
$k_{i,j}$	Eigenvalue along the edge direction of matrix $\mathbf{K}_{i,j}$
$W_g(\mathcal{T})$	Work of gravity forces
$W_p(\mathcal{T})$	Work of External Surface Pressure
$\mathbf{M}_{i,j}$	3×3 global mass matrix between vertex i and j
$\mathbf{K}^{\star}_{i,j}$	3×3 global stiffness matrix between vertex i and j that includes spring boundary conditions
$\mathbf{R}^g$	Global vector of gravity forces
$\mathbf{R}^b$	Global vector of boundary forces

References

[1] M. J. Ackerman. The visible human project. *Proceedings of the IEEE : Special Issue on Surgery Simulation*, 86(3):504–511, March 1998.

[2] N. Ayache. Epidaure: a research project in medical image analysis, simulation and robotics at INRIA. *IEEE Trans. on Medical Imaging*, October 2003. Invited Editorial.

[3] N. Ayache and H. Delingette, editors. *International Symposium on Surgery Simulation and Soft Tissue Modeling*, LNCS 2673, Juan-Les-Pins, France, jun 2003. Springer-Verlag. 386 pages.

[4] D. Baraff and A. Witkin. Large steps in cloth simulation. In ACM, editor, *Computer Graphics (SIGGRAPH'98)*, pages 43–54, Orlando (USA), July 1998.

[5] J. B. Bassingthwaighte. Strategies for the physiome project. *Annals of Biomedical Engineering*, 28:1043–1058, 2000.

[6] K-L. Bathe. *Finite Element Procedures in Engineering Analysis*. Prentice-Hall, 1982.

[7] H. Bayraktar, M. Adams, A. Gupta, P. Papadopoulos, and T. Keaveny. The role of large deformations in trabecular bone mechanical behavior. In *ASME Bioengineering Conference*, Key Biscayne, FL (USA), June 2003.

[8] Steven E. Benzley, Ernest Perry, Brett Clark Karl Merkley, and Greg Sjaardema. Comparison of all-hexahedral and all-tetrahedral finite element meshes for elastic and elasto-plastic analysis. In *4th International Meshing Roundtable*, pages 179–191. Sandia National Laboratories, October 1995.

[9] J. Besag. On the statistical analysis of dirty pictures. *Journal of Royal Statistical Society*, 48(3):326–338, 1986.

[10] D. Bielser and M. H. Gross. Interactive simulation of surgical cuts. In *Proceedings of Pacific Graphics 2000*, pages 116–125, Hong-Kong, October 2000. IEEE Computer Society Press.

[11] F. Boux de Casson and C. Laugier. Modelling the dynamics of a human liver for a minimally invasive simulator. In *Proc. of the Int. Conf. on Medical Image Computer-Assisted Intervention*, Cambridge (GB), September 1999.

[12] M. Bro-Nielsen. Finite element modeling in surgery simulation. *Proceedings of the IEEE : Special Issue on Surgery Simulation*, 86(3):490–503, March 1998.

[13] M. Bro-Nielsen and S. Cotin. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. In *Eurographics'96*, volume 3, pages 57–66, 1996.

[14] I.N. Bronshtein and K.A. Semendyayev. *Handbook of Mathematics*. Van Nostrand Reinhold Company, 1985.

[15] J. D. Brown, J. Rosen, Y. Kim, L. Chang, M. Sinanan, and B. Hannaford. In-vivo and in-situ compressive properties of porcine abdominal soft tissue. In *Medicine Meets Virtual Reality (MMVR'03)*, Newport Beach, USA, January 2003.

[16] J. Canas and F. Paris. *Boundary Element Method : Fundamentals and Application*. Oxford University Press, June 1997.

[17] F. J. Carter. Biomechanical testing of intra-abdominal soft tissue. In *International Workshop on Soft Tissue Deformation and Tissue Palpation*, Cambridge, MA, October 1998.

[18] P. G. Ciarlet. *Mathematical elasticity Vol. 1: Three-dimensional elasticity*. North-Holland, Amsterdam, 1987. ISBN 0-444-70259-8.

[19] F. Cosmi. Numerical solution of plane elasticity problems with the cell method. *Computer Methods in Engineering and Sciences*, 2(3), 2001.

[20] Ivan F. Costa and Remis Balaniuk. Lem - an approach for real time physically based soft tissue simulation. In *International Conference in Automation and Robotics (ICRA'2001)*, Seoul, May 2001.

[21] S. Cotin, H. Delingette, and N. Ayache. Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions On Visualization and Computer Graphics*, 5(1):62–73, January-March 1999.

[22] S. Cotin, H. Delingette, and N. Ayache. Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions On Visualization and Computer Graphics*, 5(1):62–73, January-March 1999.

[23] S. Cotin, H. Delingette, and N. Ayache. A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training and simulation. *The Visual Computer*, 16(8):437–452, 2000.

[24] S. Cotin, H. Delingette, J.-M. Clement, V. Tassetti, J. Marescaux, and N. Ayache. Volumetric deformable models for simulation of laparoscopic surgery. In *Proceedings of the International Symposium on Computer and Communication Systems for Image Guided Diagnosis and Therapy, Computer Assisted Radiology (CAR'96)*, volume 1124 of *International Congress Series*. Elsevier, June 1996.

[25] Couinaud. *Le foie, études anatomiques et chirurgicales*. Masson, 1957.

[26] S. A. Cover, N. F. Ezquerra, and J. F. O'Brien. Interactively Deformable Models for Surgery Simulation. *IEEE Computer Graphics and Applications*, pages 68–75, 1993.

[27] D. Dan. *Caractérisation mécanique du foie humain en situation de choc*. PhD thesis, Université Paris 7, September 1999.

[28] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. Dynamic real-time deformations using space and time adaptive sampling. *Computer Graphics Proceedings*, Aug 2001. Proceedings of SIGGRAPH'01.

[29] X. Decoret, G. Schaufler, F. Sillion, and J. Dorsey. Multi-layered impostors for accelerated rendering. In *Computer Graphics Forum 18:3 (Proceedings of Eurographics '99)*, volume 18, pages 61–73, 1999.

[30] H. Delingette. Towards realistic soft tissue modeling in medical simulation. *Proceedings of the IEEE : Special Issue on Surgery Simulation*, pages 512–523, April 1998.

[31] H. Delingette. General object reconstruction based on simplex meshes. *International Journal of Computer Vision*, 32(2):111–146, September 1999.

[32] H. Delingette and J. Montagnat. Shape and topology constraints on parametric active contours. *Journal of Computer Vision and Image Understanding*, 83:140–171, 2001.

[33] Mathieu Desbrun and Marie-Paule Gascuel. Animating soft substances with implicit surfaces. In *Computer GRAPHICS (SIGGRAPH'1995)*, Los Angeles, 1995.

[34] O. Deussen, L. Kobbelt, and P. Tucke. Using simulated annealing to obtain a good approximation of deformable bodies. In *Proc. Eurographics Workshop on Animation and Simulation*, Maastricht (NL), September 1995. springer.

[35] J. Duncan and N. Ayache. Medical image analysis: Progress over two decades and the challenges ahead. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):85–106, 2000.

[36] C. Forest, H. Delingette, and N. Ayache. Simulation of surgical cutting in a manifold mesh by removing tetrahedra,. *Medical Image Analysis*, 2003. submitted.

[37] Clément Forest, Hervé Delingette, and Nicholas Ayache. Cutting simulation of manifold volumetric meshes. In *Modelling & Simulation for Computer-aided Medicine and Surgery (MS4CMS'02)*, 2002.

[38] Clément Forest, Hervé Delingette, and Nicholas Ayache. Removing tetrahedra from a manifold mesh. In *Computer Animation (CA'02)*, pages 225–229, Geneva, Switzerland, June 2002. IEEE Computer Society.

[39] L. France, J. Lenoir, P. Meseure, and C. Chaillou. Simulation of minimally invasive surgery of intestines. In S. Richir, editor, *Fourth Virtual Reality International Conference (VRIC'2002)*, pages 21–27, 2002. ISBN 2-9515730.

[40] Laure France, Alexis Angelidis, Philippe Meseure, Marie-Paule Cani, Julien Lenoir, François Faure, and Christophe Chaillou. Implicit representations of the human intestines for surgery simulation. In *Conference on Modeling and Simulation for Computer-aided Medicine and Surgery (MS4CMS'02)*, Rocquencourt, Novembre 2002.

[41] Y. C. Fung. *Biomechanics - Mechanical Properties of Living Tissues.* Springer-Verlag, second edition, 1993.

[42] G. Picinbono and H. Delingette and N. Ayache. Non-linear and anisotropic elastic soft tissue models for medical simulation. In *ICRA2001: IEEE International Conference Robotics and Automation*, Seoul Korea, May 2001. Best conference paper award.

[43] S. Gibson, J. Samosky, A. Mor, C. Fyock, E. Grimson, T. Kanade, R. Kikinis, H. Lauer, and N. McKenzie. Simulating arthroscopic knee surgery using volumetric object representations, real-time volume rendering and haptic feedback . In J. Troccaz, E. Grimson, and R. Mosges, editors, *Proceedings of the First Joint Conference CVRMed-MRCAS'97*, volume 1205 of *Lecture Notes in Computer Science*, pages 369–378, March 1997.

[44] E. Gladilin. *Biomechanical Modeling of Soft Tissue and Facial Expressions for Craniofacial Surgery Planning.* PhD thesis, Freie Univerisität Berlin (Germany), October 2002.

[45] Stefan Gottschalk, Ming Lin, and Dinesh Manocha. Obb-tree: A hierarchical structure for rapid interference detection. *Proceedings of SIGGRAPH 96*, pages 171–180, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

[46] R. Hodgskinson and J.D. Currey. Young modulus, density and material properties in cancellous bone over a large density range. *Journal of Materials Science: Materials in Medicine*, 3:377–381, 1992.

[47] J. C. Houbolt. A recurrence matrix solution for the dynamic response of elastic aircraft. *Journal of Aeronautical science*, 17:540–550, 1950.

[48] J. D. Humphrey, R. K. Strumpf, and F. C. P. Yin. Determination of a Contitutive Relation for Passive Myocardium: I. A New Functional Form. *ASME Journal of Biomechanical Engineering*, 112:333–339, August 1990.

[49] J. D. Humphrey and F. C. P. Yin. On Constitutive Relations and Finite Deformations of Passive Cardiac Tissue: I. A Pseudostrain-Energy Function. *ASME Journal of Biomechanical Engineering*, 109:298–304, November 1987.

[50] P. Hunter and A. Pullan. *FEM/BEM Notes.* University of Auckland, New-Zeland, 1997. available at http://www1.esc.auckland.ac.nz/Academic/Texts/fembemnotes.pdf.

[51] Berci G. Hunter J.G., Sackier J.M. Training in laparoscopic cholecystectomy : Quantifying the learning curve. *journal of Endoscopic Surgery*, 8:28–31, 1994.

[52] D. L. James and D. K. Pai. Artdefo accurate real time deformable objects. In *Computer Graphics (SIGGRAPH' 1999)*, pages 65–72, 1999.

[53] B. Joe. Geompack – a software package for the generation of meshes using geometric algorithms. *Journal of Advanced Eng. Software*, 13:325–331, 1991.

[54] M. Kaiss and P. Le Tallec. La modélisation numérique du contact œil-trépan. *Revue Européenne des éléments Finis*, 5(3):375–408, 1996.

[55] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–331, 1988.

[56] M. Kauer, V. Vuskovic, J. Dual, Gabor Székely, and M. Bajka. Inverse finite element characterization of soft tissues. In *4th Int. Conf. on Medical Image Computing and Computer-Assisted Intervention (MICCAI'01)*, volume 2208 of *LNCS*, pages 128–136, Utrecht, October 2001.

[57] J. Kaye, F. Primiano, and D. Metaxas. A 3d virtual environment for modeling mechanical cardiopulmonary interactions. *Medical Image Analysis (Media)*, 2(2):1–26, 1997.

[58] Dave Knott and Dinesh Pai. Collision and interference detection in real-time using graphics hardware. In *Proceedings of Graphics Interface*, Halifax, Canada, June 2003.

[59] Ch. Kuhn, U. Kühnapfel, H.-G. Krumm, and B. Neisius. A 'virtual reality' based training system for minimally invasive surgery. In *Proc. Computer Assisted Radiology (CAR '96)*, pages 764–769, Paris, June 1996.

[60] U. Kuhnapfel, H. akmak, and H. Maa. Endoscopic surgery training using virtual reality and deformable tissue simulation. In *Computers and Graphics, 24:671–682, 2000.*, 2000.

[61] Z. Liu and L. E. Bilston. On the viscoelastic character of liver tissue: experiments and modelling of the linear behaviour. *Biorheology*, 37:191–201, 2000.

[62] Z. Liu and L. E. Bilston. Large deformation shear properties of liver tissue. *Biorheology*, 39:735–742, 2002.

[63] J.-C. Lombardo, M.-P. Cani, and F.Neyret. Real-time collision detection for virtual surgery. In *Computer Animation*, Geneva Switzerland, may 1999.

[64] Jean-Christophe Lombardo, Marie-Paule Cani, and Fabrice Neyret. Real-time Collision Detection for Virtual Surgery . In *Computer Animation*, pages 82–89, Geneva - Switzerland, May 1999.

[65] W. Lorensen and H.E. Cline. Marching cubes: a high resolution 3d surface construction algorithm. *ACM Computer Graphics (SIGGRAPH'87)*, 21:163–169, 1987.

[66] Jean Louchet, Xavier Provot, and David Crochemore. Evolutionary identification of cloth animation model. In *Workshop on Computer Animation and Simulation (Eurographics'95)*, pages 44–54, 1995.

[67] A Lumsdaine and J Siek. The Matrix Template Library, 1998. `http://www.lsc.nd.edu/research/mtl/`.

[68] R. H Macmillan. A new method for the numerical evaluation of determinants. *J. Roy. Aeronaut. Soc.*, 59(772), 1955.

[69] A. Manduca, R. Muthupillai, P. Rossman, J. Greenleaf, and L. Ehman. Visualization of tissue elasticity by magnetic resonance elastography. In *Proc of Visualization in Biomedical Imaging (VBC'96)*, pages 63–68, Hamburg, Germany, 1996.

[70] B. Marcus. Hands on : Haptic feedback in surgical simulation. In *Proc. of Medecine Meets Virtual Reality IV (MMVR IV)*, pages 134–139, San Diego, CA, January 1996.

[71] William Mark, Scott Randolph, Mark Finch, James Van Verth, and Russell M. Taylor II. Adding force feedback to graphics systems: Issues and solutions. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 447–452. ACM SIGGRAPH, Addison Wesley, August 1996.

[72] W. Maurel, Y. Wu, and N. Magnenat Thalmann D. Thalmann. *Biomechanical Models for Soft Tissue Simulation*. ESPRIT Basic Research Series. Springer-Verlag, 1998.

[73] Cesar Mendoza Serrano and Christian Laugier. Realistic haptic rendering for highly deformable virtual objects. In *Proc. of the Int. Conf. on Virtual Reality*, Yokohama (JP), March 2001.

[74] K. Miller. Constitutive modelling of abdominal organs. *Journal of Biomechanics*, 33(3):367–373, 2000.

[75] J. Montagnat and H. Delingette. Globally constrained deformable models for 3d object reconstruction. *Signal Processing*, pages 173–186, 1998.

[76] A. Nava, E. Mazza, F. Kleinermann, N. Avis, and J. McClure. Determination of the mechanical properties of soft human tissues through aspiration experiments. In *Proc. of Conference on Medical Robotics, Imaging And Computer Assisted Surgery: MICCAI 2003*, LNCS, Montreal, Canada, November 2003.

[77] N. M. Newmark. A method of computation for structural dynamics. *Journal of Engineering Mechanics Division*, 85:67–94, 1959.

[78] A. O'Mahony, J. Williams, and J. Katz. Anisotropic elastic properties of cancellous bone from a human edentulous mandible. In *Proc. of ASME Bioengineering'99 Conference*, 1999.

[79] Steve Owen. A survey of unstructured mesh generation technology. Technical report, Department of Civil and Environmental Engineering, Carnegie Mellon University, 2000.

[80] X. Papademetris, P. Shi, D. P. Dione, A. J. Sinusas, R. T. Constable, and J. S. Duncan. Recovery of soft tissue object deformation from 3d image sequences using biomechanical models. In *XVI-th International Conference on Information Processing In Medical Imaging, IPMI'99*, pages 352–357, Visegrád, Hungary, June 28 - July 2 1999.

[81] V.N. Parthasarathy, C.M. Graichen, and A.F. Hathaway. A comparison of tetrahedron quality measures. *Finite Elements in Analysis and Design*, 15:255–261, 1993.

[82] G. Picinbono, H. Delingette, and N. Ayache. Non-Linear Anisotropic Elasticity for Real-Time Surgery Simulation. *Graphical Models*, 65(5):305–321, September 2003.

[83] G. Picinbono, J.-C. Lombardo, H. Delingette, and N. Ayache. Anisotropic Elasticity and Forces Extrapolation to Improve Realism of Surgery Simulation. In *ICRA2000: IEEE International Conference Robotics and Automation*, pages 596–602, San Francisco USA, April 2000.

[84] G. Picinbono, J-C. Lombardo, H. Delingette, and N. Ayache. Improving realism of a surgery simulator: linear anisotropic elasticity, complex interactions and force extrapolation. *Journal of Visualisation and Computer Animation*, 13(3):147–167, july 2002.

[85] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C.* Cambridge Press, 1991.

[86] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in FORTRAN: The Art of Scientific Computing 2nd ed.* Cambridge University Press, Cambridge, England, 1992.

[87] M. A. Puso and J. A. Weiss. Finite Element Implementation of Anisotropic Quasi-linear Viscoelasticity Using a Discrete Spectrum Approximation. *ASME Journal of Biomechanical Engineering*, 120(1), February 1998.

[88] M. Putti and C. Cordes. Finite element approximation of the diffusion operator on tetrahedra. *SIAM Journal of Scientific Computing*, 19(4):1154–1168, July 1998.

[89] A. Quarteroni, M. Tuveri, and A. Veneziani. Computational vascular fluid dynamics: problems, models and methods. *Computing and Visualization in Science, 2 (2000), pp. 163–197.*, 2000.

[90] A. Radetzky. The simulation of elastic tissues in virtual medicine using neuro-fuzzy systems. In *Medical Imaging 98: Image Display*, San Diego, CA, February 1998.

[91] Youssef Saad. *Iterative Methods for Sparse Linear Systems.* WPS, 1996.

[92] I. Sakuma, Y. Nishimura, C. Kong Chui, E. Kobayashi, H. Inada, X. Chen, and T. Hisada. In vitro measurement of mechanical properties of liver tissue under compression and elongation using a new test piece holding method with surgical glue. In *International Symposium on Surgery Simulation and Soft Tissue Modeling*, number 2673 in LNCS, pages 284–292, Juan-Les-Pins, France, June 2003. Springer-Verlag.

[93] R. Satava. 1994 medicine meets virtual reality conference proceedings. In *Medicine 2001: The King Is Dead*, 1994.

[94] R. Satava. Medical virtual reality : The current status of the future. In *Proc. of 4th conf.Medecine Meets Virtual Reality (MMVR IV)*, pages 100–106, 1996.

[95] W. J. Schroeder, J. Zarge, and W. Lorensen. Decimation of triangles meshes. In *Computer Graphics (SIGGRAPH'92)*, volume 26. ACM, August 1992.

[96] M. Sermesant, Y. Coudière, H. Delingette, and N. Ayache. Progress towards an electro-mechanical model of the heart for cardiac image analysis. In *IEEE International Symposium on Biomedical Imaging (ISBI'02)*, pages 10–14, 2002.

[97] M. Sermesant, O. Faris, F. Evans, E. McVeigh, Y. Coudière, H. Delingette, and N. Ayache. Preliminary validation using in vivo measures of a macroscopic electrical model of the heart. In N. Ayache and H. Delingette, editors, *International Symposium on Surgery Simulation and Soft Tissue Modeling (IS4TM'03)*, volume 2673 of *Lecture Notes in Computer Science*. Springer-Verlag Heidelberg, 2003.

[98] Francois X. Sillion, George Drettakis, and Benoit Bodelet. Efficient impostor manipulation for real-time visualization of urban scenery. In *Proceedings of Eurographics'97*, Budapest, Hungary, September 1997.

[99] Simail: product of Simulog S.A. - 1, rue James Joule - 78286 Guyancourt Cedex - France, http://www.simulog.fr.

[100] Z. Soferman, D. Blythe, and N. John. Advanced graphics behind medical virtual reality : Evolution of algorithms, harware and software interfaces. *Proceedings of the IEEE : Special Issue on Surgery Simulation*, 86(3):531–554, March 1998.

[101] L. Soler, H. Delingette, G. Malandain, J. Montagnat, N. Ayache, J.-M. Clément, C. Koehl, O. Dourthe, D. Mutter, and J. Marescaux. Fully automatic anatomical, pathological and fonctionnal segmentation from ct-scans for hepatic surgery. In *Medical Imaging 2000*, San Diego, February 2000.

[102] L. Soler, G. Malandain, and H. Delingette. Segmentation automatique : application aux angioscanners 3d du foie. *Traitement du signal*, 15(5):411–431, 1998. in French.

[103] A.J.M. Spencer. *Deformations of Fibre-Reinforced Materials*. Clarendon Press, Oxfordn, 1972.

[104] A.J.M. Spencer. *Continuum Theory of Fiber-Reinforced Composites*. Springer-Verlag, New York, 1984.

[105] G. Szekely, M.Baijka, and C. Brechbuhler. Virtual reality based simulation for endoscopic gynaecology. In *proceedings of Medicine Meets Virtual Reality (MMVR'99)*, pages 351–357, San Francisco (USA), 1999.

[106] M. Teschner, B. Heidelberger, M. Muller, D. Pomeranets, and M. Gross. Optimized spatial hashing for collision detection of deformable objects. In *Proc. Vision, Modeling, Visualization VMV'03*, Munich, Germany, November 2003.

[107] M. Vidrascu, H. Delingette, and N. Ayache. Finite element modeling for surgery simulation. In *First MIT Conference on Computational Fluid and Solid Mechanics*, 2001.

[108] Vlachos, Peters, Boyd, and Mitchell. Curved pn triangles. In *2001 ACM Symposium on Interactive 3D Graphics*, 2001.

[109] Jeffrey A. Weiss, John C. Gardiner, and Krista M. Quapp. Material Models for the Study of Tissues Mechanics. *Proc International conference on Pelvic and Lower Extrimity Injuries*, pages 249–261, December 1995. Wash DC.

[110] Mason Woo, Jackie Neider, and Tom Davis. *OpenGL Programing Guide*. Addison-Wesley, 1997.

[111] Y. Yamashita and M. Kubota. Ultrasonic characterization of tissue hardness in the in-vivo human liver. In *Proc. of IEEE Ultrasonics Symposium*, pages 1449–1453, 1994.

[112] Denis Zorin, Peter Schroeder, and Wim Sweldens. Interpolating subdivision for meshes with arbitrary topology. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 189–192. ACM Press, 1996.