Connective Tissues Simulation on GPU

Julien Bosman, Christian Duriez, Stéphane Cotin

Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2013)
J. Bender, J. Dequidt, C. Duriez, and G. Zachmann (Editors)

Connective Tissues Simulation on GPU

Julien Bosman, Christian Duriez, Stéphane Cotin

Inria Lille Nord Europe - Lille 1 University, France

Abstract

Recent work in the field of medical simulation have led to real advances in the mechanical simulation of organs. However, it is important to notice that, despite the major role they may have in the interaction between organs, the connective tissues are often left out of these simulations. In this paper, we propose a model which can rely on either a mesh based or a meshless methods. To provide a realistic simulation of these tissues, our work is based on the weak form of continuum mechanics equations for hyperelastic soft materials. Furthermore, the stability of deformable objects simulation is ensured by an implicit temporal integration scheme. Our method allows to model these tissues without prior assumption on the dimension of their of their geometry (curve, surface or volume), and enables mechanical coupling between organs. To obtain an interactive frame rate, we develop a parallel version suitable for to GPU computation. Finally we demonstrate the proper convergence of our finite element scheme.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling —Physically based modeling

1. Introduction

Surgical simulation based on virtual-reality techniques and its application to training has been a subject of research for the last two decades. Yet, the current commercial simulators still suffer from a lack of realism. It mainly comes from the difficulties to accurately simulate the deformations of soft-tissues in real-time. Some recent results allow to improve both precision and performance of simulations while offering the opportunity to integrate patient-specific data (mostly issued from medical imaging) [DPP12]. As a result new applications like pre-operative planning or assisting tools for surgeons can be envisioned.

The common requirement for these applications is to improve precision and interactivity by both enhancing biomechanical models and increasing the efficiency of computations. A majority of recent work deals with specific organs like the liver [MHC*10], the brain [HWM07] or even the pelvic system [RRD*11]. One of the main difficulties in these models is to impose proper boundary conditions.

So far, very little work has been dedicated to the simulation of the interactions between organs. In [CJA*10], these interactions are simulated as frictional contacts and solved using an optimized contact resolution method. Though it constitutes a real advance by its ability to simulate multiple interacting organs, this model is not completely realistic. Indeed, organs are rarely directly in contact. Some of them are surrounded and connected by fibers known as connective tissues or *fasciae*, which are fibrous biological tissues comprising tendons and ligaments. The liver, for example is linked to the stomach by the hepatogastric ligament and to the diaphragm by the falciform ligament. However, when these interactions are simulated using frictional contacts, these organs slide on each other whereas they should be, at least partially, linked by these connective tissues.

One of the difficulties in the modeling of these tissues resides in the fact that they don't have uniform geometrical properties. Since they fill the volume between certain organs, they can exhibit both volumetric and very thin parts for the same anatomical structure. Creating volumetric meshes for such 3D domains can be difficult: to accurately restitute thin parts of a model, meshing would require using small elements. Thus, it would lead to a higher number of elements and/or ill-conditioning problems due to bad shaped elements, increasing the computational cost. Moreover, some of the connective tissues may have complex geometries, they can appear as an heterogeneous volume of soft tissues constituted by a random arrangement of fibers shaped like a dense web. They can contain small void parts and not fill entirely the volume they occupy. Thus, they cannot be con-

sidered as a totally continuous volume of matter. For these reasons, they are often ignored in the simulations. Yet, they play a major role by influencing boundary conditions of the surrounding organs.

Hence, connective tissues need a simulation method offering more liberty on the discretization of the domain and being able to impose proper boundary conditions for the simulation of the organs. In addition, a low computational cost is required, to not decrease the performance of the simulation of the organs, but while still keeping accurate results.

In this paper, to improve the realism of simulations, we develop a numerical method able to simulate connective tissues and the mechanical coupling they provide between organs. For this purpose, we propose an approach in which connective tissues are modeled as an interface simulated with six degrees of freedom (DOF) mechanical points. This approach relies on a *total Lagrangian* formulation and can be used as either a mesh based or meshfree method. Our work enables mechanical coupling between organs as well as the propagation of boundary conditions between organs which are not directly in contact. Mechanical coupling is made easier in that we use the same type of degrees of freedom as a rigid objects, which can be used to simulate bones, but also shell and beam elements, that can be used to simulate hollow organs (like bladder, stomach ...), and ligaments. Thus, our method offers a wide range of mechanical coupling. In order to achieve real time performance, we propose a parallel implementation on GPU.

1.1. Previous work

In the field of the finite element method, the handling of large deformations due to geometrical non-linearities (*i.e.* local rotations), has been proposed by Felippa in [Fel00] with the *corotational* method. By this method, the motion of the elements is decomposed into a rigid rotation and a deformation. In [GW08], Georgii *et al.* proposed a corotational method based on an energy minimization formulation to extract both motions, associated with a multigrid solver. The modeling of hyperelasticity is a subject of major importance in the field of the surgical simulation seeing that most of biological tissues exhibit this property. In [MHC*10] Marchesseau *et al.* propose an hyperelastic model for liver simulation. The total Lagrangian approach is quite commonly used in the simulation of elastic objects as in [MJLW07] – where Miller *et al.* use this method to simulate organs modeled with mesh objects – or as in [HWJM10] where Horton *et al.* use a total Lagrangian approach along with a meshless method. Both of these work rely on an explicit time integration. However, even though they are quite fast, explicit time integrators exhibit several disadvantages, especially instability issues. Very small time steps, especially with stiff or heterogeneous materials and detailed meshes, are required to maintain stability [BW98].

The use of nodes with six degrees of freedom has been proposed several times and is commonly encountered in the simulation of beams and shells. The Cosserat's theory [CC09], which is extensively used for beams and shells, is based upon a 6-DOF description of the deformation field. However, their use for the simulation of volumetric deformable objects is less common. In [MHTG05] Müller *et al.* use a method known as *shape matching* where degrees of freedom are partitioned in several groups. For each of these groups, an optimal rotation and a translation are estimated with respect to the rest configuration and a set of springs move the vertices to their rest position. This idea is also used in [CM05], the difference being that Chentanez *et al.* use a set of oriented particles along with a generalized shape matching to animate deformable models. However, their methods does not rely on physical laws and, although they handle rotations, they are still based on 3-DOF nodes. Mixed mesh based/meshfree approaches have also been proposed. An hybrid method has been used by Sifakis et al in [SSIF07] where the volumetric mesh of simulated objects embeds a point cloud to ease the handling of collisions and fractures by avoiding objects remeshing.

GPUs are increasingly used in physically-based simulations because they enable the simulation of more detailed models and faster computations. In [ACF11], Allard *et al.* proposed a simulation of deformable bodies on GPU using tetrahedral meshes relying on an implicit time integration. In [DGW11] Dick *et al.* proposed a GPU-based approach of a finite element method relying on hexahedra used along with a multigrid solver. A review of the use of GPU-based approaches in surgical simulation is given in [SM06].

Connective tissues simulation is briefly treated in [Sis12] but only from a collision handling point of view. To the best of our knowledge, the simulation of the mechanical behavior of connective tissues has never been treated.

Our method is similar to [GBFP11], in which Gilles *et al.* propose a method based on *dual quaternions* to simulate deformable objects according to the continuum mechanics. Originally designed to simulate very sparse deformable models, this method offered a limited convergence. However, using simple quaternion (instead of dual quaternion which increases the number of DOF per node) along with a massively parallel implementation, it is now possible to use many more DOF while keeping the simulation at interactive frame rate. In addition, by using nodes with six degrees of freedom, our method is directly compatible with models that use the same type of degrees of freedom like rigid bodies and shell or beam elements, which makes mechanical coupling between these models easier.

2. Simulation

2.1. Kinematics

The objects are discretized by two kinds of primitive: a set of samples, used to discretize the objects volume, and a set

of frames. Each frame $\mathbf{q}_i$ has six degrees of freedom (3 in translation and 3 in rotation):

$$\mathbf{q}_i = [x_1\ x_2\ x_3\ r_1\ r_2\ r_3]^t = [\mathbf{x}_i\ \mathbf{r}_i]^t \tag{1}$$

where $\mathbf{x}_i$ is the position and $\mathbf{r}_i$ the rotation vector corresponding to the orientation of the frame i (in practice, the rotation part is coded using quaternions). Samples have only 3 DOF. Considering that we use a *total Lagrangian* formulation, the global position of a sample is expressed with respect to its local position in each frame coordinate system, in the rest (undeformed) configuration. Thus, the position of the samples is entirely controlled by their surrounding frames. The position of a sample s locally to a frame i in the rest configuration is given by:

$$\bar{\mathbf{l}}_s^i = \bar{\mathbf{R}}_i^t\ [\bar{\mathbf{p}}_s - \bar{\mathbf{x}}_i] \tag{2}$$

where $\bar{\mathbf{R}}_i$ and $\bar{\mathbf{x}}_i$ are respectively the rotation matrix and the position of frame i, and $\bar{\mathbf{p}}_s$ is the sample's coordinates in the rest configuration. The coordinates of a sample s in the deformed configuration is given by:

$$\mathbf{p}_s = \sum_i w_i(\bar{\mathbf{p}}_s)\left(\mathbf{x}_i + \mathbf{R}_i\,\bar{\mathbf{l}}_s^i\right) \tag{3}$$

with $w_i(\cdot)$ being the value of the shape function associated with the sample s and the frame i, and $\mathbf{x}_i$ and $\mathbf{R}_i$ the coordinates and the rotation matrix corresponding to the orientation of the frame i in the deformed configuration. The shape functions are used to interpolate the displacements values – which are only known at the frames – at the samples location. With a mesh based scheme, shape functions generally rely on barycentric coordinates of the elements. In a meshfree approach, many shape functions can be used, like the MLS as in [BLG94], compactly supported radial basis functions as in [LJ95] or in the SPH [DC96] or even *Natural Neighbors Interpolation*.

Depending on whether the method is used with a mesh based or meshfree scheme, the sampling method varies. With a mesh based scheme, the samples are placed inside the elements of the mesh using barycentric coordinates. Multiple samples can be placed for each element. The frames are placed at each vertex of the geometric mesh. With a meshfree scheme, a boundary surface mesh is used to provide the geometrical domain of the tissues. Using a voxelization of this mesh, two different samplings are generated using a *Poisson distribution* [Bri07] based on the inter-particle distance. The first one is used as frames coordinates whereas the second one, with a smaller inter-particle distance, is used as samples coordinates. Note that initial the frames orientation has no influence on the computations as we only track the differences between undeformed and current configurations. For this reason, the frames are initially aligned with world axis.

2.2. Dynamics

2.2.1. Internal forces

This approach is based upon continuum mechanics which establishes the relationship between a displacement field (*i.e.* deformations experienced at each point of the domain) and the associated elastic forces, according to the following scheme:

$$\underbrace{\mathbf{u}_i}_{\text{displacement}} \rightarrow \underbrace{\mathbf{F}_i}_{\substack{\text{deformation}\\ \text{gradient}}} \rightarrow \underbrace{\mathbf{L}_i}_{\substack{\text{strain}\\ \text{tensor}}} \rightarrow \underbrace{\mathbf{S}_i}_{\substack{\text{stress}\\ \text{tensor}}} \rightarrow \underbrace{\mathbf{f}_i}_{\substack{\text{internal}\\ \text{force}}}$$

During the simulation, internal forces are only applied on the frames. The samples are used to compute the deformation gradient, as well as strain and stress tensors. The calculation of the frame's internal forces is achieved by integration on the volume of the object. Thus, the force to apply on a frame is calculated by accumulating the stress of the neighboring samples.

In this article we use a total Lagrangian formulation, in which all the displacements, at any time during the simulation, are evaluated with respect to the rest configuration. The total Lagrangian approach is well adapted to elastic objects which always retrieve their rest shape. The calculation of the internal forces $\mathbf{f}$ relies on a variational formulation (*i.e.* a weak form similar to the *virtual work theorem*):

$$\delta W = \delta\mathbf{q}^t \cdot \mathbf{f} = \delta\mathbf{q}^t \int_\Omega \frac{\partial \mathbf{L}^t}{\partial \mathbf{q}}\{\mathbf{S}\}d\Omega \tag{4}$$

with

$$\mathbf{L} = \frac{1}{2}(\mathbf{F}^t\mathbf{F} - \mathbf{I}) \tag{5}$$

where δW represents the work's variation, $\mathbf{L}$ is the *Green-Lagrange* strain tensor and $\mathbf{F}$ is the deformation gradient. $\{\mathbf{S}\}$ is the second *Piola-Kirchhoff* (PK2) stress tensor in *Voigt* notation (see [BLM00], appendix 1) and Ω is the domain of the object. As we are considering nonlinear elasticity (large deformations), we choose the Green-Lagrange strain tensor, which is widely used for modeling hyperelasticity in a total lagrangian formulation. This choice is pertinent in that connective tissues with thin parts are likely to experience large deformations with geometrical nonlinearities (local rotations). Thus, the rotational invariance of this tensor makes a good fit for our purpose. The strain tensor derivative with respect to the degrees of freedom is given by:

$$\frac{\partial \mathbf{L}}{\partial \mathbf{q}} = \frac{1}{2}\left(\frac{\partial \mathbf{F}^t}{\partial \mathbf{q}}\mathbf{F} + \mathbf{F}^t\frac{\partial \mathbf{F}}{\partial \mathbf{q}}\right) \tag{6}$$

The first step in the internal forces computation is to calculate the deformation gradient for each sample, *i.e.* the spatial derivative of displacements $\mathbf{F}^s = \nabla\mathbf{p}_s$. Then, by looking at equation (2) one can notice that:

$$\frac{\partial \bar{\mathbf{l}}_s^i}{\partial \bar{\mathbf{x}}} = \frac{\partial(\bar{\mathbf{R}}_i^t\ [\bar{\mathbf{p}}_s - \bar{\mathbf{x}}_i])}{\partial \bar{\mathbf{x}}} = \bar{\mathbf{R}}_i^t \tag{7}$$

Thus, the deformation gradient at a sample s, which is a 3×3 matrix, is obtained by differentiating equation (3) and by using the result of equation (7):

$$\mathbf{F}^s = \frac{\partial \mathbf{p}_s}{\partial \bar{\mathbf{x}}} = \sum_i \left[\nabla w_{is} \left(\mathbf{x}_i + \mathbf{R}_i \bar{\mathbf{l}}_s^i \right)^t + w_{is} \mathbf{R}_i \bar{\mathbf{R}}_i^t \right] \quad (8)$$

with $w_{is} = w_i(\bar{\mathbf{p}}_s)$.

In order to calculate the Green-Lagrange strain tensor derivative of equation (6), the deformation gradient has to be differentiated with respect to $\mathbf{q}$. By doing so, one can obtain the variation of deformation gradient:

$$\delta \mathbf{F}^s = \frac{\partial \mathbf{F}^s}{\partial \mathbf{q}} \delta \mathbf{q} \quad (9)$$

Those 9 values are expressed with respect to an infinitesimal variation of both the position and the orientation of the frames $\delta\mathbf{q}$, *i.e.* with respect to $\delta\mathbf{x}$ and $\delta\mathbf{r}$ ($\delta\mathbf{q}^t = [\delta\mathbf{x}\ \delta\mathbf{r}]^t$). Thus, for each coefficient i, j of $\delta\mathbf{F}^s$ one obtains:

$$\begin{aligned} \delta \mathbf{F}^s_{ij} = \sum_n \frac{\partial w_{ns}}{\bar{\mathbf{x}}_{,j}} \delta \mathbf{x}_n^i - \Bigg(\frac{\partial w_{ns}}{\bar{\mathbf{x}}_{,j}} \left[\mathbf{R}_n \bar{\mathbf{l}}_s^n{}^{\wedge} \right]^{i,} + \\ w_{ns} \left[\mathbf{R}_n \left[\bar{\mathbf{R}}_n^t \right]^{,j\ \wedge} \right]^{i,} \Bigg) \cdot \delta \mathbf{r_n} \end{aligned} \quad (10)$$

where lowercase indices correspond to a spatial coordinate and n identifies a frame.

$\mathbf{R}_n\bar{\mathbf{l}}_s^n$ corresponds to $\bar{\mathbf{l}}_s^n$ rotated by $\mathbf{R}_n$, $[\mathbf{v}^{\wedge}]$ is the antisymmetric-matrix representation of the cross product, $[\mathbf{A}]^{i,}$ and $[\mathbf{A}]^{,j}$ are respectively the i^{th} row and the j^{th} column of the matrix $\mathbf{A}$.

By rewriting the 9 values of $\delta\mathbf{F}^s$ as a column vector, the equation (10) can be expressed as a matrix-vector product for each interacting frame/sample pair so that $w_{is} > 0$:

$$\begin{bmatrix} \delta \mathbf{F}^s_{11} \\ \delta \mathbf{F}^s_{12} \\ \vdots \\ \delta \mathbf{F}^s_{33} \end{bmatrix} = \sum_i \mathbf{B}^{i,s}\, \delta \mathbf{q}_i \quad (11)$$

with $\delta\mathbf{q}_i^t = [\delta\mathbf{x}_i\ \delta\mathbf{r}_i]^t$. The matrix $\mathbf{B}$, given in appendix, is a 9×6 matrix.

Having the derivative of the deformation gradient, the strain tensor derivative $\partial\mathbf{L}^{i,s}/\partial\mathbf{q}$, can be calculated. One can use equations (6) and (11) to express $\delta\mathbf{L}^{i,s} = \partial\mathbf{L}^{i,s}/\partial\mathbf{q}$ as a matrix vector product. The resulting vector is a 6-dimensional vector since $\delta\mathbf{L}^{i,s}$ is a symmetric matrix. Hence, for $\delta\mathbf{L}^{i,s}_{jk}$ with $j = 1$ and $k = 2$:

$$\begin{aligned} \delta \mathbf{L}_{12} &= \left([\delta \mathbf{F}^t]^{1,} \cdot [\mathbf{F}]^{,2} + [\mathbf{F}^t]^{1,} \cdot [\delta \mathbf{F}]^{,2} \right) \\ &= \left([\delta \mathbf{F}_{11}\ \delta \mathbf{F}_{21}\ \delta \mathbf{F}_{31}] \cdot [\mathbf{F}]^{,2} + [\mathbf{F}^t]^{1,} \cdot [\delta \mathbf{F}_{12}\ \delta \mathbf{F}_{22}\ \delta \mathbf{F}_{32}] \right) \end{aligned} \quad (12)$$

Furthermore, from equation (10):

$$[\delta \mathbf{F}_{11}\ \delta \mathbf{F}_{21}\ \delta \mathbf{F}_{31}] = \left[[\mathbf{B}]^{1,} \cdot \delta \mathbf{q}_i\ [\mathbf{B}]^{4,} \cdot \delta \mathbf{q}_i\ [\mathbf{B}]^{7,} \cdot \delta \mathbf{q}_i \right] \quad (13)$$

and

$$[\delta \mathbf{F}_{12}\ \delta \mathbf{F}_{22}\ \delta \mathbf{F}_{32}] = \left[[\mathbf{B}]^{2,} \cdot \delta \mathbf{q}_i\ [\mathbf{B}]^{5,} \cdot \delta \mathbf{q}_i\ [\mathbf{B}]^{8,} \cdot \delta \mathbf{q}_i \right] \quad (14)$$

substituting equations (13) and (14) into equation (12) and factoring by $\delta\mathbf{q}_i$ one can obtain:

$$\begin{aligned} \delta \mathbf{L}_{12} = \Big[&\mathbf{F}_{12}[\mathbf{B}]^{1,} + \mathbf{F}_{22}[\mathbf{B}]^{4,} + \mathbf{F}_{32}[\mathbf{B}]^{7,} \\ &+ \mathbf{F}_{11}[\mathbf{B}]^{2,} + \mathbf{F}_{21}[\mathbf{B}]^{5,} + \mathbf{F}_{31}[\mathbf{B}]^{8,} \Big] \cdot \delta \mathbf{q}_i \end{aligned} \quad (15)$$

Then, rewriting the 6 values of $\delta\mathbf{L}^{i,s}$ in Voigt notation one finally has:

$$\begin{bmatrix} \delta \mathbf{L}^{i,s}_{11} \\ \delta \mathbf{L}^{i,s}_{22} \\ \vdots \\ 2\delta \mathbf{L}^{i,s}_{12} \end{bmatrix} = \sum_i \mathbf{M}^{i,s}\, \delta \mathbf{q}_i \quad (16)$$

where each line of $\mathbf{M}^{i,s}$ has the same form as the equation (15).

By substituting equation (16) into equation (4) and then dividing by $\delta\mathbf{q}$ one obtains the internal force to be applied on a frame:

$$\mathbf{f} = \int_\Omega \mathbf{M}^t \{\mathbf{S}\} d\Omega \quad (17)$$

However, equation (17) is given for the continuous case, meaning that this integral should be evaluated on the whole domain covered by the object. To enable the computation of internal forces, this equation has to be converted into a discrete form. To do so, samples will be used, as they discretize the object's volume. Thus, the discrete expression of the internal force to be applied at a frame i is given by:

$$\mathbf{f}_i = -\sum_s \left[\mathbf{M}^{i,s} \right]^t \{\mathbf{S}^s\} v_s \quad (18)$$

where v_s is the volume associated to the sample s and $\mathbf{S}^s$ in Voigt notation, is its stress tensor. The volume of the samples should be chosen so that the sum of the samples volumes equals the total volume of the object. The model used for stress computation is an hyperelastic model known as the *Saint Venant - Kirchhoff* model. This model generalizes the *Hooke's law* to large deformations. However, our formulation is compatible with more advanced constitutive laws. The expression of the PK2 stress tensor is given by:

$$\mathbf{S} = \lambda\, tr(\mathbf{L})\mathbf{I} + 2\mu\mathbf{L} \quad (19)$$

where λ and μ are the Lamé coefficients which depend on the material to be simulated.

2.2.2. Implicit integration

Solving force equation is a non-linear problem which requires the tangent matrix linearization in order to be solved. Once it is linearized, the problem is reduced to multiple

linear problems. These linear problems are then solved using a *conjugate gradient* solver which is a *Newton-Raphson* like method. Thanks to the implicit time integration, larger time steps can be used while keeping a stable simulation. The linearization of internal forces means that they have to be differentiated with respect to the frame's degrees of freedom. This amounts to expressing the variation of the force according to an infinitesimal variation of stress. From [BLM00](§6.4.2) the internal force derivative can be separated in two parts:

$$\delta\mathbf{f}^{int} = \delta\mathbf{f}^{mat} + \delta\mathbf{f}^{geo} \tag{20}$$

where $\delta\mathbf{f}^{mat}$ is the *material tangent stiffness* and $\delta\mathbf{f}^{geo}$ is the *geometric stiffness*. The expression of the material tangent stiffness is given by:

$$\delta\mathbf{f}^{mat} = \int_{\Omega} \mathbf{M}^{\mathbf{t}}\{\delta\mathbf{S}\}d\Omega \tag{21}$$

with $\{\delta\mathbf{S}\}$ being the stress tensor derivative in Voigt notation:

$$\{\delta\mathbf{S}\} = \mathbf{C}\{\delta\mathbf{L}\} \tag{22}$$

where $\mathbf{C}$ is the matrix associated to the Hooke's law. As in the previous section, this equation has to be discretized to be integrated, resulting in:

$$\delta\mathbf{f}_i^{mat} = -\sum_s \mathbf{M}^{i,s}\,\Delta\mathbf{S}^s v_s \tag{23}$$

with:

$$\Delta\mathbf{S}^s = \sum_i \mathbf{C}\mathbf{M}^{i,s}\,\delta\mathbf{q}_i \tag{24}$$

According to the standard product differentiation rule, the expression of the geometric stiffness is given by:

$$\delta\mathbf{f}^{geo} = \int_{\Omega} \delta\mathbf{M}^t\{\mathbf{S}\}d\Omega \tag{25}$$

or in the discrete case:

$$\delta\mathbf{f}_i^{geo} = -\sum_s \left[\delta\mathbf{M}^{i,s}\right]^t \{\mathbf{S}^s\} v_s \tag{26}$$

with $\delta\mathbf{M} = \partial\mathbf{M}/\partial\mathbf{q}$.

By looking at equation (26), one can notice that the matrix $\mathbf{M}$ has to be differentiated with respect to $\mathbf{q}$. To better illustrate the process of this differentiation, we can examine the last component of the geometric stiffness (*i.e.* $\delta\mathbf{f}^{geo}$) as an example:

$$\mathbf{f}_{i_6}^{geo} = -\sum_s \left[\delta\mathbf{M}^{i,s}\right]^{,6} \cdot \{\mathbf{S}^s\} v_s \tag{27}$$

In other words, the 6$^{\text{th}}$ component of the geometric stiffness is the dot product of the 6$^{\text{th}}$ column of $\delta\mathbf{M}$ by $\mathbf{S}$. Without loss of generality, the i^{th} component of the geometric stiffness is the dot product of the i^{th} column of $\delta\mathbf{M}$ and $\mathbf{S}$. Developing equation (27) leads to:

$$\begin{aligned}\mathbf{f}_{i_6}^{geo} = -\sum_s \big[&\mathbf{S}_1^s(\delta\mathbf{F}_{11}\mathbf{B}_{16} + \mathbf{F}_{11}\delta\mathbf{B}_{1,6} + \cdots \\ &+ \mathbf{S}_6^s(\cdots + \delta\mathbf{F}_{31}\mathbf{B}_{86} + \mathbf{F}_{31}\delta\mathbf{B}_{86})\big] v_s \cdot \delta\mathbf{q}_i\end{aligned} \tag{28}$$

with $\delta\mathbf{B}_{ij} = \partial\mathbf{B}_{ij}/\partial\mathbf{q}$. One can notice that equation (28) involves the deformation gradient derivatives, which are already known by evaluating equation (11), but also the derivatives of the $\mathbf{B}$ matrix coefficients. Let $\delta\mathbf{B}_{16} = [\delta\mathbf{B}_{16}^{\mathbf{x}}\ \delta\mathbf{B}_{16}^{\mathbf{r}}]$, the concatenation of two 3-dimensional vectors. Since the derivative of all the coefficients of the right half of $\mathbf{B}$ have the same form, we can differentiate $\delta\mathbf{B}_{16}^{\mathbf{r}}$ as an example:

$$\delta\mathbf{B}_{16}^{\mathbf{r}} = -\frac{\partial w}{\partial x}\delta\left(\mathbf{R}_i\,\bar{\mathbf{l}}_s^i\right)_y - w\delta\left(\mathbf{R}_i[\bar{\mathbf{R}}_i^t]^{,x}\right)_y \tag{29}$$

where $(\mathbf{R}_i\bar{\mathbf{l}}_s^i)_x$ is the x component of the vector $\mathbf{R}_i\,\bar{\mathbf{l}}_s^i$. Proceeding with the differentiation results in:

$$\begin{aligned}\delta\mathbf{B}_{16} &= -\frac{\partial w}{\partial x}\left[-\mathbf{R}[\bar{\mathbf{l}}_s^i\wedge]\right]^{2,} - w\left[-\mathbf{R}_i\left[[\bar{\mathbf{R}}_i^t]^{,x}\wedge\right]\right]^{2,} \\ &= \frac{\partial w}{\partial x}\left[\mathbf{R}[\bar{\mathbf{l}}_s^i\wedge]\right]^{2,} + w\left[\mathbf{R}_i\left[[\bar{\mathbf{R}}_i^t]^{,x}\wedge\right]\right]^{2,}\end{aligned} \tag{30}$$

and $\delta\mathbf{B}_{16}^{\mathbf{x}}$ is a null 3-dimensional vector.

The differentiation all the coefficients of the left half of the $\mathbf{B}$ matrix results in a null vector. Moreover, on the right half, the differentiation of nine coefficients results in a null vector. Because the right half of the $\mathbf{B}$ is built using antisymmetric matrices (see appendix), some of the coefficients on that half are merely the opposite of one another (for instance, $\delta\mathbf{B}_{16} = -\delta\mathbf{B}_{74}$). As a result, the right half only includes nine unique derivatives.

3. GPU implementation

The GPU implementation of this method is achieved using CUDA. Before the simulation is run, during the initialization step, the shape function values are precomputed thanks to the total Lagrangian formulation. Indeed, forces are computed with respect to the rest configuration, thus, so are the shape functions. These data are stored in structures, which are described in the next subsection.

In this section, the data structures needed by computations are first introduced. Then we describe force and force derivative computations. At last, we discuss some concepts of this implementation.

3.1. Data structures

Expressing the problem using a total Lagrangian formulation allows us to precompute the values of shape functions. These values are stored in a table containing the following data: the value and the gradient of the shape function at each neighboring frame as well as their index. In the same manner, a reverse table is built. This second structure, shown on figure (1) is an indirection table presenting, for a given frame, the

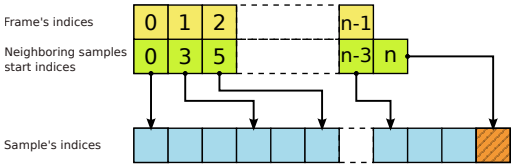

Figure 1: Reverse indirection table.

number of neighboring samples (*i.e.* interacting samples) it has, as well as their indices. The aim of this second table is to enable the summation of the forces produced by the stress experienced by the neighboring samples.

3.2. Force and force derivative computation

Forces and force derivatives are computed in a similar manner. In the first part of the algorithm, we generate a set of partial forces (as well as partial force derivatives) contributions. In other words, we compute forces and force derivatives individually for each interacting sample/frame pair: as a sample experiences a stress, it produces a force on each of its neighboring frames. In the second part of the algorithm, we sum all these contributions to produce one force (force derivative) vector per frame.

In this method, there are two entities on which computations have to be performed: samples and frames. In the first part of this implementation, the parallelization is done on the sample level. By this, we mean that all the computations on the samples are done in parallel.

During the force computation step, we assign one thread per sample. The deformation gradient is first computed and then stored in registers as well as the Saint Venant-Kirchhoff stress tensor. Then for each frame the sample interacts with, a partial force is computed by multiplying the transpose of the **M** matrix with the stress tensor. However, the implementation relies on a matrix-free approach: neither the **B** nor the **M** matrix is explicitly built in the device global memory (VRAM). Instead, the result of the multiplication is evaluated row by row so that we only need one matrix row at a time. Thus, due to its small size, this matrix row can be stored in registers. The resulting partial force vector is stored in shared memory and then transferred to a temporary buffer contained in the device global memory.

The computation of the force derivative is based on the same approach as that of the force computations. However, in order to evaluate these derivatives, the computation of the stress tensor derivative of equation (24) is required. For each sample treated in parallel, the stress tensor derivative is computed by iterating sequentially through its neighboring frames. The force derivative computation is also done in a matrix-free way. The resulting force derivative is stored in shared memory and then transferred to the global memory.

The first part of both forces and forces derivatives computation produces a buffer of partial force (derivative) contributions made of 6-dimensional vectors. In order to compute the total force (derivative) on each frame individually, these partial contributions have to be summed. The second part of this implementation relies on a segmented inclusive scan [SHZO07]. This time, computations are parallelized on the frame level, *i.e.* each frame is treated in parallel. We assign 6 threads to each frame (one per vector component), in groups of 96 threads. Using the reverse indirection table, partial force (derivative) contributions of the neighboring samples are stored in shared memory and summed using a parallel scan operation. When this is done, the resulting force (derivative) vector is transferred to global memory.

The whole algorithm is depicted on figure (2). The first kernel is dedicated to partial forces (derivatives) computations. Each thread treats one sample and produces a partial contribution for each of its neighboring frame. In the second kernel, one frame is treated by 6 thread which use the indirection table to pick and sum the partial contributions of each neighboring samples.

3.3. Details of the implementation

The parallelization of the computations on the sample level has several advantages. First, because there are more samples than frames, we are able to provide with the GPU more work to parallelize, increasing its occupancy and thus its efficiency. Then, many of the computations have to be done on the samples, as shown in table (1). By doing so, we are able to compute these quantities only once. Since each computation on a sample is independent from other samples, there is no need for synchronization barriers. On the other hand, because frames are influenced by multiple samples, the computation of frames forces would result in concurrent write conflicts. Using atomic operations is not a viable option since it reduces significantly the efficiency because writes are serialized. Besides, atomic operations on floats are not supported on older hardware. Thus the computation of frames forces as a set of partial contributions is required. Using this approach computations with only two kernels – one for the partial contributions computations and a second dedicated to the summation of the later – can be performed. By having only two kernels per computation, we limit the effect of kernel launching overhead.

A great attention has been given to memory management. First, the use of a matrix-free approach is of major importance in our problem. Certain quantities like the **B** and **M** matrices have to be computed for each interacting frame/sample pair. Storing all these matrices would require a substantial amount of memory. Since global memory access is one of the main bottlenecks in GPU computing, this would degrade the performance. Thus, a matrix-free approach makes it possible to reduce the memory consumption and the memory access overhead. Hence, storing partial contributions in shared memory before storing them

Samples	**F**, **L**, **S**, Δ**S**, δ**F**
Frames	**f**, δ**f**
Sample/Frame pairs	**B**, **M**

Table 1: Quantities that have to be computed and the entities that rely on

in global memory allows us to perform coalesced memory writes. On graphic hardware, the memory is divided in segments. If all threads in a group access addresses that fall into the same segment (*i.e.* accesses are coalesced), memory operations can be performed in one transaction, limiting memory access overhead. However, because of the limited amount of shared memory available on GPUs, our implementation is restricted to use at most 16 neighboring frames per sample. Though, this limitation is not a major concern since 16 neighbors per samples are enough for most applications. Indeed, when used with a mesh based scheme, our method needs either 4 neighbors (for tetrahedral linear interpolation) or 8 neighbors (hexahedral trilinear interpolation). With a meshfree scheme, this restriction is not really a limiting factor since our models are sparse.

The summation of partial we assign 6 threads per frame and choose a group size of 96 threads. We choose this group size because it is a multiple of the warp size (32 threads) which is the number of threads running concurrently on a GPU multiprocessor. By doing so, we ensure that there are no inactive threads in a group (except in the last one).

4. Results

4.1. Convergence

In order to assess the accuracy of the mechanical model, we made a comparative test involving the cantilever beam. We choose to compare the behavior of two beams simulated with our method using tetrahedral linear and hexahedral trilinear interpolation on one side, with two beams simulated with the standard finite element method on the other. The first one is modeled using tetrahedra whereas the second is modeled using hexahedra. These two beams are simulated using a corotational formulation included in the SOFA framework. This analysis is focused on the number of degrees of freedom required to converge toward the solution. The simulated beam is anchored at one end and the only external force experienced is due to its own weight. Its dimensions are $16 \times 1 \times 1$ meters, its weight is 5 kg, its Young modulus is 40000 Pa and its Poisson ratio is 0.3. We voluntarily choose a highly deformable beam in order to study the convergence in the case of large deformations.

For convergence analysis, multiple simulations have been performed with different resolutions. The chart presented on figure (3) shows the deflection of the free end of the beam, at equilibrium, varying with respect to the number of degrees

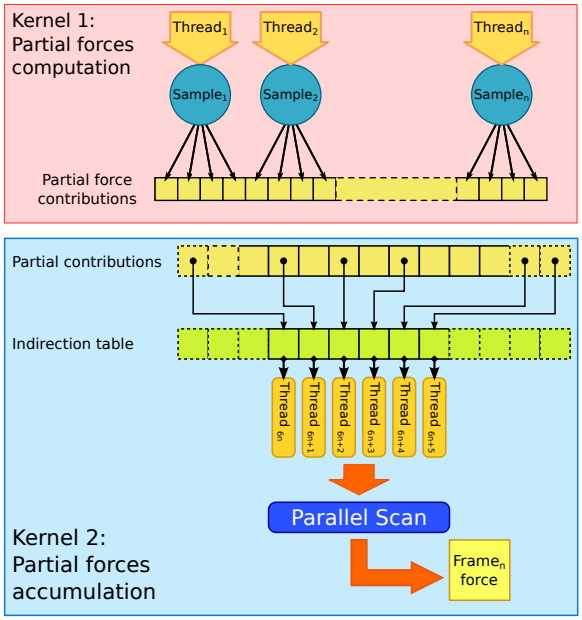

Figure 2: GPU algorithm. The first pass is done on the samples and produces a set of partial contributions (kernel 1). The second pass is done on the frames to sum partial contributions.

of freedom for each method. It clearly demonstrates that the frame method converges faster than finite element method. By this, we mean that the frame method needs fewer degrees of freedom to converge toward the same solution. For example, a beam which length and section are respectively discretized with 20 and 9 frames gives the same results as a beam modeled with $64 \times 4 \times 4$ hexahedra. These results allow us to highlight some advantages of this method. First, since it requires less degrees of freedom, the final system is faster to solve. In addition, it seems that the frame method – using tetrahedral linear, or hexahedral trilinear shape functions – is less sensitive to the element conditioning. Indeed, it is possible to use much elongated tetrahedra and hexahedra, whereas classical finite element method requires tetrahedra or hexahedra which are as regular as possible.

Figure (4) shows the result of two beams simulated with the frame method (right) and with the hexahedral finite element method (left). The beam modeled with frames needs fewer frame nodes to provide similar results.

4.2. Coupling

The proposed method enables mechanical coupling thanks to its ability to share its degrees of freedom (*i.e.* the frames) with another mechanical model simulated with different methods also using 6 degrees of freedom per node. Among these methods, we can find the beams elements [DCLN06]

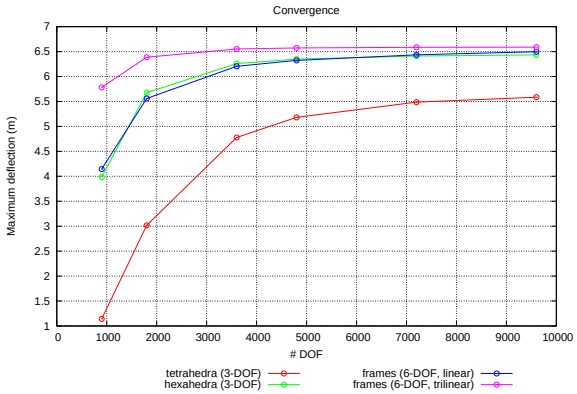

Figure 3: Convergence analysis of the frame method.

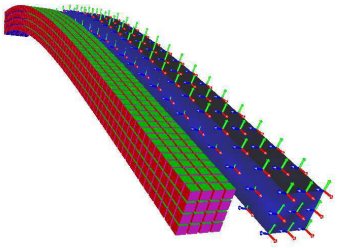

Figure 4: A beam modeled the the frame method (left) and with hexahedra (right). The error is below 2%.

or shells elements [CDC10] that can be used to model hollow organs like stomach or bladder. Rigid bodies also use 6 degrees of freedom per node and can be used to simulate bones and articulations. Thanks to our technique, it is possible to quickly model connective tissues between organs or other anatomical structures in two steps. First, the region corresponding to the connective tissues has to be defined by placing frames and samples inside. Then the frames that should be shared between the connective tissues and the connected organs have to be defined. Figure (5) illustrates this mechanism: connective tissues (yellow) have been sampled with frames (red) and samples. The surfaces of the top and bottom organs (pink) have been modeled with frames (blue). The organs and connective tissues frames inside the green area are coupled and considered as one mechanical object, allowing for propagation of boundary conditions. Coupling with a rigid object is slightly different, as rigid bodies do not have frames on their surface, but only one frame located at their center of mass. To do so, the frames of the connective tissues of the interface (*i.e* which lie on the rigid surface) have a rigid motion w.r.t. to the rigid body. Figure (6) illustrates the ability of the presented method to couple multiple objects. In these two simulations, the frames of the front and rear edges of the topmost surface are fixed, all the other frames are able to move freely.

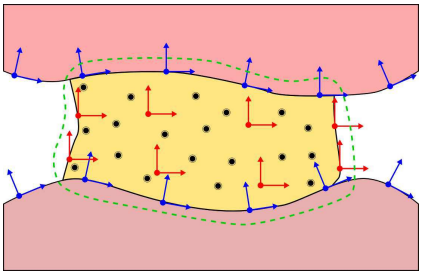

Figure 5: Mechanical coupling between two organs. The frames of the lower and the upper surface (blue) inside the green area are shared with the frames of the connective tissues (red).

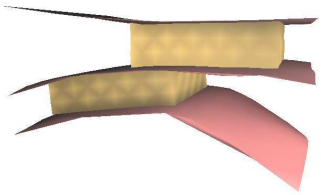

Figure 6: Coupling between multiple objects.

Simulations of mechanical coupling using more complex models have also been realized. Figure (7) shows the result of a simulation in which an uterus is linked to a bladder using connective tissues. The top of the uterus surface is fixed. The bladder and the uterus are simulated using shell elements. The connective tissues are discretized with 22 frames and 827 samples.

When using a meshless shape function, the method makes the coupling of objects of different resolutions easier, thanks to the unstructured property of meshless shape functions. Indeed, the number of neighboring frames, in contrast with a mesh based method, is not limited and is generally influenced by the radius of influence of the shape function. Moreover, a meshless scheme is interesting in the case of connective tissues with thin parts in that it avoids meshing.

4.3. Performance

Table (2) shows the timings obtained during the tests of the non-threaded CPU implementation. The computer used for these tests is equipped with a 2.67 GHz Intel Xeon W3520 CPU. The following parameters have been observed: the number of frames, the number of samples, as well as the number of neighboring frames per sample. It clearly highlights that samples and frames counts are the parameters that affects the performance the most. It also shows that the method rapidly becomes unusable for real-time applications as the number of samples increase.

Table (3) presents the results of a comparative perfor-

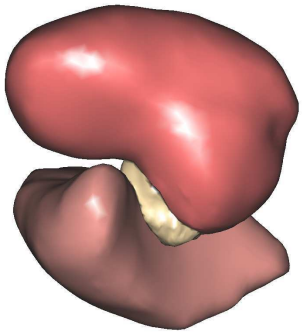

Figure 7: Simulation of the pelvic system. The uterus (top) is linked to the bladder (bottom) by a layer of fasciae.

Test	Frame	Samples	Neighbors	Time (ms).
Beam	72	400	4	32,65
Beam	72	400	8	45,42
Beam	72	400	6	55,15
Beam	72	400	10	69,56
Beam	72	500	4	39,18
Beam	72	864	4	43,42
Beam	72	1470	4	114,3
Beam	72	2340	4	160,53
Beam	144	750	4	54,49
Beam	288	750	4	56,64
Beam	500	750	4	58,5
Beam	792	750	4	69,12
Coupling	160	400	4	53,22

Table 2: CPU timings.

mance test between CPU and GPU implementation. Those tests were achieved using a conjugate gradient solver performing exactly 60 iterations per time step. Each sample has 4 neighboring frames. However, it should be emphasized that CG converge with a number of iterations that depends linearly on the number of degrees of freedom. Moreover CGs may converge slowly in some cases (heterogeneous material for example). To overcome this problem, it is possible to use some preconditioning techniques to improve the convergence rate. The device used during these tests is a Nvidia GTX 570 with 1280 Mb of global memory. The data gathered in this table are: the number of samples, the time elapsed for a single time step with the CPU and the GPU implementations, and the acceleration factor. The timings are given for a full time step which includes internal/external forces computations and system solving. Visual rendering is not taken into consideration. The result shows an interesting performance gain even for a relatively small number of samples. They also show that beyond 2600 samples, the implementation scales well with the number of samples. With the GPU implementation more complex models can be simulated at interactive frame rates.

Samples.	Time (CPU)	Time (GPU)	Accel.
800	78,42	15,25	5,14
2600	251,77	17,11	14,72
6070	595,77	23,39	25,47
12700	1228,8	32,29	38,07
16210	1561,8	39,29	39,75
19990	1990,06	43,12	46,15
26750	2675,47	53,09	50,39
34610	3468,54	66,1	52,47

Table 3: Comparison between CPU and GPU performance (timing are in ms).

5. Conclusion and future work

We presented a method for the simulation of connective tissues that can be either mesh based or meshfree. Our method is able to simulate volumetric, but also very thin objects and relies on a total Lagrangian formulation. The implicit temporal integration allows for stable simulation, even with larger time steps. The frame method on GPU is appropriate for connective tissues simulation in that it combines the need for fewer degrees of freedom, which speeds-up the computation, and the performance of the GPU implementation. As a result, we are able to propose simulations with interactive frame rate which both enables the simulation of complex interactions between the organs and other anatomical structures, and leaves enough computational resources for organs simulation. This work can be improved on several ways. First, by using standard shape functions commonly encountered in mesh based or meshfree methods, we made the assumption that connective tissues are a continuous volume of matter, which does not reflect the true nature of these tissues. To improve the realism of our model, we have to model their discontinuities. This could be done by using more advanced shape functions like the ones used in the *extended finite element method* (XFEM). Then, one of the most interesting prospects is a solver based on domain decomposition. Currently the connected organs and connective tissues are merged into one system of equations. With the domain decomposition we could simulate each connected organs and connective tissues independently in a first phase, and compute interactions between them in a second phase. As far as performance is concerned, the GPU implementation may also be improved. Considering it is impossible to perform only coalesced access, improvements in the data locality could limit the loss of performance due to uncoalesced accesses, by taking advantage of the cache of the last generation of GPUs.

Appendices

A. B matrix

For a frame i and a sample s, the $\mathbf{B}$ matrix is given by:

$$\begin{bmatrix}
\frac{\partial w_{is}}{\partial x} & 0 & 0 & - \left[\frac{\partial w_{is}}{\partial x}[\mathbf{R}_i \cdot \overline{\mathbf{l}}_s^{\,i} \wedge]^{x,} + [\mathbf{R}_i \cdot [\overline{\mathbf{R}}_i^{\,t}]^{,x} \wedge]^{x,}\, w_{is}\right] \\
\frac{\partial w_{is}}{\partial y} & 0 & 0 & - \left[\frac{\partial w_{is}}{\partial y}[\mathbf{R}_i \cdot \overline{\mathbf{l}}_s^{\,i} \wedge]^{x,} + [\mathbf{R}_i \cdot [\overline{\mathbf{R}}_i^{\,t}]^{,y} \wedge]^{x,}\, w_{is}\right] \\
\frac{\partial w_{is}}{\partial z} & 0 & 0 & - \left[\frac{\partial w_{is}}{\partial z}[\mathbf{R}_i \cdot \overline{\mathbf{l}}_s^{\,i} \wedge]^{x,} + [\mathbf{R}_i \cdot [\overline{\mathbf{R}}_i^{\,t}]^{,z} \wedge]^{x,}\, w_{is}\right] \\
0 & \frac{\partial w_{is}}{\partial x} & 0 & - \left[\frac{\partial w_{is}}{\partial x}[\mathbf{R}_i \cdot \overline{\mathbf{l}}_s^{\,i} \wedge]^{y,} + [\mathbf{R}_i \cdot [\overline{\mathbf{R}}_i^{\,t}]^{,x} \wedge]^{y,}\, w_{is}\right] \\
0 & \frac{\partial w_{is}}{\partial y} & 0 & - \left[\frac{\partial w_{is}}{\partial y}[\mathbf{R}_i \cdot \overline{\mathbf{l}}_s^{\,i} \wedge]^{y,} + [\mathbf{R}_i \cdot [\overline{\mathbf{R}}_i^{\,t}]^{,y} \wedge]^{y,}\, w_{is}\right] \\
0 & \frac{\partial w_{is}}{\partial z} & 0 & - \left[\frac{\partial w_{is}}{\partial z}[\mathbf{R}_i \cdot \overline{\mathbf{l}}_s^{\,i} \wedge]^{y,} + [\mathbf{R}_i \cdot [\overline{\mathbf{R}}_i^{\,t}]^{,z} \wedge]^{y,}\, w_{is}\right] \\
0 & 0 & \frac{\partial w_{is}}{\partial x} & - \left[\frac{\partial w_{is}}{\partial x}[\mathbf{R}_i \cdot \overline{\mathbf{l}}_s^{\,i} \wedge]^{z,} + [\mathbf{R}_i \cdot [\overline{\mathbf{R}}_i^{\,t}]^{,x} \wedge]^{z,}\, w_{is}\right] \\
0 & 0 & \frac{\partial w_{is}}{\partial y} & - \left[\frac{\partial w_{is}}{\partial y}[\mathbf{R}_i \cdot \overline{\mathbf{l}}_s^{\,i} \wedge]^{z,} + [\mathbf{R}_i \cdot [\overline{\mathbf{R}}_i^{\,t}]^{,y} \wedge]^{z,}\, w_{is}\right] \\
0 & 0 & \frac{\partial w_{is}}{\partial z} & - \left[\frac{\partial w_{is}}{\partial z}[\mathbf{R}_i \cdot \overline{\mathbf{l}}_s^{\,i} \wedge]^{z,} + [\mathbf{R}_i \cdot [\overline{\mathbf{R}}_i^{\,t}]^{,z} \wedge]^{z,}\, w_{is}\right]
\end{bmatrix}$$

References

[ACF11] ALLARD J., COURTECUISSE H., FAURE F.: Implicit fem and fluid coupling on gpu for interactive multiphysics simulation. In *ACM SIGGRAPH 2011 Talks* (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 52:1–52:1. 2

[BLG94] BELYTSCHKO T., LU Y. Y., GU L.: Element-free Galerkin methods. *International Journal for Numerical Methods in Engineering 37*, 2 (1994), 229–256. 3

[BLM00] BELYTSCHKO T., LIU W. K., MORAN B.: *Nonlinear Finite Elements for Continua and Structures*. Wiley, 2000. 3, 5

[Bri07] BRIDSON R.: Fast Poisson disk sampling in arbitrary dimensions. *ACM SIGGRAPH* (2007), 2006. 3

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 43–54. 2

[CC09] COSSERAT E., COSSERAT F.: *Théorie des corps déformables*. 1909. 2

[CDC10] COMAS O., DURIEZ C., COTIN S.: Shell Model for Reconstruction and Real-Time Simulation of Thin Anatomical Structures. In *MICCAI (2)* (China, 2010), pp. 371–379. 8

[CJA*10] COURTECUISSE H., JUNG H., ALLARD J., DURIEZ C., LEE D. Y., COTIN S.: GPU-based Real-Time Soft Tissue Deformation with Cutting and Haptic Feedback. *Progress in Biophysics and Molecular Biology 103*, 2-3 (2010), 159–168. 1

[CM05] CHENTANEZ N., MÜLLER M.: Solid Simulation with Oriented Particles. 2

[DC96] DESBRUN M., CANI M.: Smoothed particles: A new paradigm for animating highly deformable bodies. *Proceedings of the Eurographics workshop on Computer animation and simulation* (1996), 61–76. 3

[DCLN06] DURIEZ C., COTIN S., LENOIR J., NEUMANN P.: New approaches to catheter navigation for interventional radiology simulation. *Computer aided surgery : official journal of the International Society for Computer Aided Surgery 11*, 6 (Nov. 2006), 300–8. 7

[DGW11] DICK C., GEORGII J., WESTERMANN R.: A real-time multigrid finite hexahedra method for elasticity simulation using CUDA. *Simulation Modelling Practice and Theory 19*, 2 (2011), 801–816. 2

[DPP12] DERAM A., PAYAN Y., PROMAYON E.: Towards a generic framework for evaluation and comparison of soft tissue modeling. In *Proceedings of the 19th Medicine Meets Virtual Reality Conference* (2012), pp. 116–122. 1

[Fel00] FELIPPA C. A.: *A systematic approach to the element-independent corotational dynamics of finite elements.* Tech. Rep. CU-CAS-00-03, Center for aerospace structures, january 2000. 2

[GBFP11] GILLES B., BOUSQUET G., FAURE F., PAI D.: Frame-based Elastic Models. *ACM Transactions on Graphics 30*, 2 (Apr. 2011). 2

[GW08] GEORGII J., WESTERMANN R.: Corotated finite elements made fast and stable. In *VRIPHYS* (2008), pp. 11–19. 2

[HWJM10] HORTON A., WITTEK A., JOLDES G. R., MILLER K.: A meshless Total Lagrangian explicit dynamics algorithm for surgical simulation. *Integration The Vlsi Journal* (2010), 977–998. 2

[HWM07] HORTON A., WITTEK A., MILLER K.: Subject-specific biomechanical simulation of brain indentation using a meshless method. In *Proceedings of the 10th international conference on Medical image computing and computer-assisted intervention - Volume Part I* (Berlin, Heidelberg, 2007), MICCAI'07, Springer-Verlag, pp. 541–548. 1

[LJ95] LIU W., JUN S.: Reproducing kernel particle methods. *International Journal for Numerical Methods in Fluids 20*, 8-8 (1995), 1081–1106. 3

[MHC*10] MARCHESSEAU S., HEIMANN T., CHATELIN S., WILLINGER R., DELINGETTE H.: Fast porous visco-hyperelastic soft tissue model for surgery simulation: Application to liver surgery. *Progress in Biophysics and Molecular Biology / Progress in Biophysics & Molecular Biology 103*, 2-3 (2010), 185–196. 1, 2

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. *ACM Trans. Graph. 24*, 3 (July 2005), 471–478. 2

[MJLW07] MILLER K., JOLDES G., LANCE D., WITTEK A.: Total lagrangian explicit dynamics finite element algorithm for computing soft tissue deformation. *Communications in Numerical Methods in Engineering 23*, 2 (2007), 121–134. 2

[RRD*11] RIVAUX G., RUBOD C., DEDET B., BRIEU M., GABRIEL B., DE LANDSHEERE L., DEVOS P., DELMAS V., COSSON M.: Biomechanical characterisation of uterine ligaments. implications for the pelvic floor. *Pelvi-Perineologie 6*, 2 (2011), 67–74. 1

[SHZO07] SENGUPTA S., HARRIS M., ZHANG Y., OWENS J. D.: Scan Primitives for GPU Computing. In *GRAPHICS HARDWARE 2007* (2007), Association for Computing Machinery, pp. 97–106. 6

[Sis12] SISMANIDIS E.: Real-time simulation of blood vessels and connective tissue for microvascular anastomosis training. In *Virtual Reality Short Papers and Posters (VRW), 2012 IEEE* (2012), pp. 109–110. 2

[SM06] SØRENSEN T., MOSEGAARD J.: An introduction to GPU accelerated surgical simulation. *Proceedings of the 3rd international conference on Biomedical Simulation* (2006), 93–104. 2

[SSIF07] SIFAKIS E., SHINAR T., IRVING G., FEDKIW R.: Hybrid simulation of deformable solids. *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), 81–90. 2