HAL
open science

Corotated meshless implicit dynamics for deformable bodies

Jean-Nicolas Brunet, Vincent Magnoux, Benoît Ozell, Stéphane Cotin

To cite this version:

Jean-Nicolas Brunet, Vincent Magnoux, Benoît Ozell, Stéphane Cotin. Corotated meshless implicit dynamics for deformable bodies. WSCG 2019 - 27th International Conference on Computer Graphics, Visualization and Computer Vision, May 2019, Pilsen, Czech Republic. hal-02119414v1

HAL Id: hal-02119414

https://inria.hal.science/hal-02119414v1

Submitted on 26 Jun 2019 (v1), last revised 7 Oct 2019 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Corotated meshless implicit dynamics for deformable bodies

Jean-Nicolas Brunet
Inria Nancy Grand-Est
jnbrunet2000@gmail.com

Vincent Magnoux
École polytechnique de Montréal
vincent.magnoux@polymtl.ca

Benoît Ozell
École polytechnique de Montréal
benoit.ozell@polymtl.ca

Stéphane Cotin
Inria Nancy Grand-Est
stephane.cotin@inria.fr

ABSTRACT

This paper proposes a fast, stable and accurate meshless method to simulate geometrically non-linear elastic behaviors. To address the inherent limitations of finite element (FE) models, the discretization of the domain is simplified by removing the need to create polyhedral elements. The volumetric locking effect exhibited by incompressible materials in some linear FE models is also completely avoided. Our approach merely requires that the volume of the object be filled with a cloud of points. To minimize numerical errors, we construct a corotational formulation around the quadrature positions that is well suited for large displacements containing small deformations. The equations of motion are integrated in time following an implicit scheme. The convergence rate and accuracy are validated through both stretching and bending case studies. Finally, results are presented using a set of examples that show how we can easily build a realistic physical model of various deformable bodies with little effort spent on the discretization of the domain.

Keywords

Meshless methods; Physically-based modelling; Interactive simulation; Point-based models

1 INTRODUCTION

Simulating realistic deformations of a soft virtual object is a complex task that can quickly transform into a considerable challenge when the simulation needs to be performed in a close to real-time framerate. Starting from a 3D representation of a deformable object, the model must account for all external forces such as gravity, collision impacts and other elements that alter the surface of the object. This can be achieved by building a volumetric representation of the object and by deriving the displacement field according to the continuum mechanics laws in a discretized space. This approach is usually referred to as a *physics-based simulation* framework [NMK*06]. Hence, the object's deformation results from the natural equilibrium between external forces being captured by the simulation process and elastic properties (mainly the resistance to stretching and compression) of the object's material. The complexity of the method then greatly depends on the application for which the simulation process is designed and more importantly, the performance criteria to be achieved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

1.1 Four key performance criteria

Over the last decade, a large number of engineering and medical computer simulation applications have been developed. For this type of application, the *accuracy* of the solution is usually the main performance criterion imposed on simulation methods. Simply stated, the main concern here is *how close is the virtual representation to the object's real deformation*? Since this strongly depends on the physical laws governing the simulation model, this condition can also be translated as *how do the states of deformation compare to a theoretical solution*? Physics-based methods are usually best suited where high accuracy is required.

Other applications such as interactive modelers or virtual simulators require quick updates of the various deformation states. This brings two other performance criteria into play, namely the *speed* and the *stability* of the simulation. In this work, the speed constraint is expressed as either a real-time requirement (around 30 frames per second) or a close to real-time (around 1 frame per second) requirement. The stability constraint, on the other hand, relates to the robustness of the simulation process and its response to unexpected external forces.

The *simplicity* of the simulation framework is our fourth and last criterion. The objective here is to adequately correlate the accuracy of the solution produced by the simulation process to the amount of configuration work and technical knowledge required by the end-user.

To be effective, the simulation framework must provide an adequate balance between these four criteria. More specifically, the method must be accurate. It must also

be well suited for interactive modeling and must be flexible enough to handle the different properties of elastic materials. Finally, the model must be simple enough to allow inexperienced users to easily configure and run the simulation with minimum assistance.

1.2 Current methods

Methods based on finite elements (FE) currently form one of the most popular physics-based simulation frameworks. Using a linear FE approach and a small displacement assumption, [BC96] and [CDA99] have pioneered the field of real-time surgical simulators. To minimize the impact of rotations, [Fel00] and [NPF05] have proposed a method, called the corotational FEM, to extract a rotation matrix from the displacement of elements. Their work has opened the door to a whole new branch of FEM-based tools for interactive simulation. Unfortunately, these methods come with some drawbacks: the simulated object's volume must be pre-filled with well-formed and well-placed elements. In addition, for nearly incompressible materials, FE methods are affected by volume locking effects which lead to numerical errors that directly impact the accuracy criterion. To overcome this situation, the user must not only discretize the volume with elements, but must also make sure that these elements will not generate numerical errors. The complexity of this approach compromises another of our performance criteria, i.e. the simplicity of the framework.

To address the inherent limitations of FE models in deformable object animation, a well-established branch of solutions is gaining momentum: *meshless methods*. With these frameworks, the discretization of the domain is simplified by removing the need to create polyhedral elements. They also prevent the volumetric locking effect exhibited by incompressible materials [BLG94]. Under normal conditions, the meshless approach merely requires that the volume of the object be filled with a cloud of points, frequently called particles.

Meshless methods have already been proposed in both the Computer graphics & Animations and Computational mechanics communities. [HWJM10] has proposed a fully non-linear element-free Galerkin method [BLG94] for surgical simulation. Although no mention was made with respect to real-time compliance, all indications are that they were the first to propose a fast enough Galerkin-based solution without element-based approximations. Since then, their work has been carried over to various surgical applications [MHJW12; ZWJ*14; DRTZ16; DLLW16; WGJ*16]. While these meshless methods are accurate, their use of fully non-linear material is time-consuming and not well adapted for applications requiring a real time or close to real-time framerate. Since they use an explicit time integration scheme, they are also restricted to small and regular time step intervals which implies a less robust solution in an interactive environment.

[MKN*04] presented a completely mesh-free method based on a moving least squares (MLS) approximation and a particle density approach to integrate the elastic energy equations in space. Using the same approach, [SSP07] also presented a complete mesh-free method, but this time using a smoothed particle hydrodynamics (SPH) approximation. Unlike [MKN*04], their shape function does not rely on the inverse of a moment matrix and allows for co-linear and co-planar neighborhoods. [BIT09; PGBT18] extended this strategy by carrying the corotational principle to the SPH formulation. While these meshless methods present visually plausible results, their volumetric integration approach is based on an approximation of the particle densities. This reduces the precision of the simulation and induces instabilities. It is especially true given that their particles represent both the degrees of freedom and the integration points.

Beside physics-based simulation frameworks, geometry-based methods such as [SCL*04; MHTG05] are often proposed where only the surface representation of simulated objects matters. While these approaches produce visually pleasing results, they do not satisfy our accuracy criterion where material properties must be correctly simulated anywhere inside the simulated object.

1.3 Our proposed solution

In this paper, we propose to address the problem of 3D deformable object simulation using a meshless method whereby each object is defined only by their surface meshes. The solution borrows concepts from both the FE and meshless models. But unlike the other solutions mentioned earlier, our approach relies on two basic adjustments that are directly driven by our four performance criteria.

Firstly, unlike meshless density-based integration methods, our approach relies on background quadrature grids where the intersection between the surface mesh and grid cells are used to accurately estimate the interior domain's volume. The objective here is to both improve the integration process of the Galerkin approach and reduce numerical errors inherent to large non-linear displacements by applying the corotational principle.

Secondly, since the effects of nonlinear rotations are minimized by the corotational principle, we propose to exploit a linear approximation of the stress and strain measures. This then allows for close to real time framerates. We also propose an implicit time integration scheme for large time steps that are better suited for interactive simulations.

1.4 Paper outline

In Section 2, we present a brief overview of the mathematical framework that was selected to model material elasticity and deformation. The basic features of our proposed method are then discussed in Section 3. Among other things, we show how, from the updated positions of a set of integration points, the rotational part of a displacement can be extracted. We also expose the system assembly stages for both a static solving scheme and an implicit time integration scheme. This is followed by the introduction of a method to correctly map the vertices of the object surface mesh to the particle positions. We then explain how collision forces gathered from the surface mesh can also be mapped onto the particles. In Section 4 we present a series of case studies to measure the convergence rate and precision of our proposed model. We conclude this paper in Section 5 with some suggestions for future work.

2 MODELING ELASTIC MATERIALS

To simulate the elastic behavior of an object, a set of rules governing the shape of an object over time must be used. For a body that is stretched or compressed by external forces, whether from a gravitational field or from a collision with another object, the rules ensure that an equilibrium state is always maintained between these forces and the body's deformation resistance dictated by the intrinsic material properties. The rules also involve two key parameters to describe the stiffness of the material: i) Young's modulus E (N/m^2) which defines the lengthwise resistance to stretch and compression; and ii) Poisson's ratio ν which quantifies the perpendicular expansion (respectively the transverse compression) of the body's volume during compression (respectively stretch). A material is said to be incompressible when its Poisson's ratio approaches the limit of 0.5.

The rules that we selected for our simulation framework are derived from Cauchy's first law of motion, which states the conservation of linear momentum in a continuum. Here, Cauchy's stress tensor $\boldsymbol{\sigma}$ conveys the amount of stress (N/m^2) sustained by the material undergoing a certain deformation. Under the small strain hypothesis, if $\boldsymbol{u} = [u\ v\ w]^T$ is a displacement vector from position $\boldsymbol{x}^0$ in the undeformed state of the body to its deformed position $\boldsymbol{x} = \boldsymbol{x}^0 + \boldsymbol{u}$, the deformation of a material can then be approximated by the linear strain tensor :

$$\boldsymbol{\varepsilon}(\boldsymbol{u}) = \frac{1}{2}(\boldsymbol{\nabla u} + \boldsymbol{\nabla u}^T) \tag{1}$$

where $\boldsymbol{\nabla u}$ is the displacement gradient.

Using a constitutive model that follows Hooke's law of elasticity, we can define the stress tensor $\boldsymbol{\sigma}$ explicitly as a linear function of the displacement $\boldsymbol{u}$:

$$\boldsymbol{\sigma}(\boldsymbol{u}) = 2\mu\boldsymbol{\varepsilon} + \lambda tr(\boldsymbol{\varepsilon})\boldsymbol{I} \tag{2}$$

where $\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$ and $\mu = \frac{E}{2(1+\nu)}$ are the Lamé coefficients. It is important to note that equations 1 and 2 are linearized versions of their counterparts found in finite strain theory. This means that the stress and strain tensors can only represent relatively small and linear displacements of the body. However, they can be computed faster and, as we will see later, they can be precalculated at the beginning of the simulation. Also, we will later present a method to minimize the effects of this linearization when nonlinear transformations (rotations) are encountered.

Cauchy's first law of motion can finally be translated into a system of partial differential equations which pose our set of rules for the simulation process:

$$-\boldsymbol{\nabla}\cdot\boldsymbol{\sigma} = \boldsymbol{f} \Rightarrow \begin{cases} -(\frac{\partial\sigma_{11}}{\partial x} + \frac{\partial\sigma_{12}}{\partial y} + \frac{\partial\sigma_{13}}{\partial z}) &= f_x \\ -(\frac{\partial\sigma_{21}}{\partial x} + \frac{\partial\sigma_{22}}{\partial y} + \frac{\partial\sigma_{23}}{\partial z}) &= f_y \\ -(\frac{\partial\sigma_{31}}{\partial x} + \frac{\partial\sigma_{32}}{\partial y} + \frac{\partial\sigma_{33}}{\partial z}) &= f_z \end{cases} \tag{3}$$

where $\boldsymbol{f}$ is the external force. Finding the deformed shape of an elastic object is then reduced to the problem of solving for the unknown displacements $\boldsymbol{u}$ of the partial differential equations 3. This can be viewed as a *static simulation* as it does not involve any time integration scheme.

For a *dynamic simulation* that involves time dependent terms such as the gravitational force, equation 3 becomes:

$$\rho\frac{d^2\boldsymbol{x}}{dt^2} + \boldsymbol{f}^{\text{elastic}}(\boldsymbol{x},t) = \boldsymbol{f}^{\text{ext}}(\boldsymbol{x},t) \tag{4}$$

where ρ is the material density, $\boldsymbol{f}^{\text{elastic}}(\boldsymbol{x},t) = -\boldsymbol{\nabla\sigma}(\boldsymbol{u})$ is the internal elastic force and $\boldsymbol{f}^{\text{ext}}(\boldsymbol{x},t)$ is the external force. In this case, finding the deformed shape of an elastic object requires numerically integrating equation 4 over time.

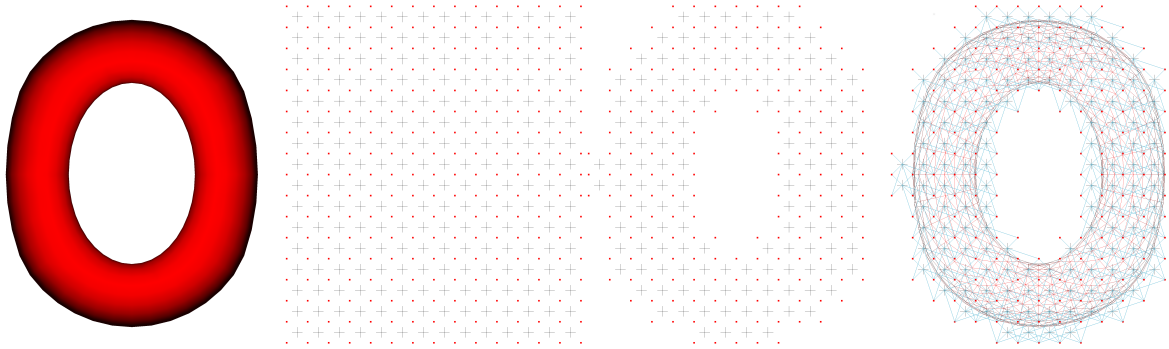
The next section presents the complete process of computing the elastic force $\boldsymbol{f}^{\text{elastic}}(\boldsymbol{x},t)$ and the description of an implicit method to integrate equation 4 over time.

3 OUR PROPOSED METHOD

In the previous section, we described a linear relationship between stress and infinitesimal strain in a *continuous domain*. In order to solve the unknown displacement field $\boldsymbol{u}$ of our object, and given that we do not have an explicit definition of it, we propose using an approach that relies on the Galerkin method, which we now outline.

3.1 The Galerkin method

The Galerkin method uses a weak formulation of the discrete partial differential equations to be solved. To simplify the reading, we temporarily disregard the time dependent terms of our equations and refer only to the



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

Figure 1: Volumetric discretizations of a 3D surface. (a) Surface mesh provided by the user. (b) Background grid where the grid's cubes are used to place the DOFs and the integration points. (c) DOFs and integration points are cropped to fit the surface mesh. (d) A neighborhood of the closest particles is built around each integration point.

static case. By multiplying equation 3 by a test function $\boldsymbol{w}$ in the Sobolev solution subspace $H^1(\Omega_0)^3$, and by following Green's theorem, the set of equations becomes

$$-\underbrace{\int_{\Omega_0} \boldsymbol{\sigma}(\boldsymbol{u}) \cdot \delta\boldsymbol{\varepsilon} d\Omega_0}_{\Pi_{elastic}} = \underbrace{\int_{\Omega_0} \boldsymbol{b} \cdot \boldsymbol{w} d\Omega_0 + \int_{\Gamma_0} \boldsymbol{t} \cdot \boldsymbol{w} d\Gamma_0}_{\Pi_{ext}} \tag{5}$$

where Ω_0 is the initial (undeformed) domain, Γ_0 is the domain boundary, $\delta\boldsymbol{\varepsilon}$ denotes the variation of the strain tensor and $\boldsymbol{t} = \boldsymbol{\sigma} \cdot \boldsymbol{n}$ is the surface normal traction vector. Here the left term $\Pi_{elastic}$ can be viewed as the internal virtual work and Π_{ext} as the work related to external loads.

3.2 Volumetric discretization: the FE approach

In finite element methods, the initial domain Ω_0 is discretized into a set of polyhedral elements (usually tetrahedrons or hexahedrons). These elements serve two important purposes: i) to construct an interpolation function of the displacement anywhere in the domain, and ii) to numerically integrate the continuous equations. The FE approach begins by building an explicit geometrical representation of an element domain Ω_e. From this representation, an interpolation $\boldsymbol{u}_e(\boldsymbol{x})$ of the displacement inside every element e with respect to its nodes is assembled. The displacement $\boldsymbol{u}(\boldsymbol{x})$ of any position $\boldsymbol{x} \in \Omega_0$ then becomes the displacement $\boldsymbol{u}_e(\boldsymbol{x})$ where element e is the one surrounding $\boldsymbol{x}$. By gathering all element nodes into a set called *degrees of freedom* (DOFs), the problem of solving for the unknown displacement field $\boldsymbol{u}(\boldsymbol{x})$ is thus reduced to one of solving for a finite vector of n unknown displacements $[\boldsymbol{u}_0, \boldsymbol{u}_1, ..., \boldsymbol{u}_n]^T$. Furthermore, placing one or more Gauss integration point inside the elements provides a means of numerically estimating the integral terms of equation 5 since elements usually conform to the surface, hence producing an accurate volumetric representation (the sum of all integration point volumes should equal the total volume of the object).

3.3 Volumetric discretization: the meshless approach

Whereas FE methods use the nodes of a polyhedral element to interpolate the displacement at Gauss points, meshless methods instead create an approximation of the displacement by considering the values of nearby points. The idea is thus to fill the interior volume of the object with an evenly distributed cloud of points, *the particles*. These particles represent the DOFs of the system and, consequently, the unknown displacements to be solved. The approximation is built by using shape functions ϕ that are evaluated at every particle near a given position. The value of the displacement $\boldsymbol{u}(\boldsymbol{x})$ becomes:

$$\boldsymbol{u} \approx \tilde{\boldsymbol{u}} = \sum_{i \in V(\boldsymbol{x})} \phi_i(\boldsymbol{x})\, \boldsymbol{u}_i \tag{6}$$

where $V(\boldsymbol{x})$ is the set of particles in the vicinity of position $\boldsymbol{x}$. Here, $\tilde{\boldsymbol{u}}$ is an approximation since our shape function does not offer the Kronecker delta property at the nodes, and is therefore not a true interpolant [BKO*96].

For the volumetric integration of equation 5, we use a background grid of regular cubic elements that completely covers the domain. The Gauss points within these volume elements, *the integration points*, are used for numerical integration of the equation. The summation of the volume of every integration point must always be very close to the total interior volume of the simulated object. The integration over the continuous domain Ω_0 becomes:

$$\int_{\Omega_0} f(\boldsymbol{u}) d\Omega_0 \approx \sum_I v_I f(\tilde{\boldsymbol{u}}_I) \tag{7}$$

where v_I is the volume of integration point I and $\tilde{\boldsymbol{u}}_I$ is the approximation of the displacement at position of I.

Using equation 6, the displacement of any integration point is estimated by accumulating the weighted displacements of the particles surrounding it. The shape functions we selected for our proposed framework are similar to those described in [HWJM10] and are found by using a moving least squares approach to minimize weighted residuals from the polynomial approximation $u^h(\boldsymbol{x}) = \boldsymbol{p}^T(\boldsymbol{x})\boldsymbol{a}(\boldsymbol{x})$. To conform with time constraints imposed by interactive simulations, we use linear basis functions $\boldsymbol{p} = [1\ x\ y\ z]$. Solving for the unknown coefficients $\boldsymbol{a}$, the shape function and its derivative become:

$$\begin{aligned} \phi_i &= \boldsymbol{P}_I \boldsymbol{A}^{-1} W_{Ii} \boldsymbol{P}_i \\ \phi_{i,x} &= [\boldsymbol{P}_{I,x} \boldsymbol{A}^{-1} W_{Ii} + \boldsymbol{P}_I (\boldsymbol{A}_{,x}^{-1} W_{Ii} + \boldsymbol{A}^{-1} W_{Ii,x})] \boldsymbol{P}_i \end{aligned} \quad (8)$$

where $\boldsymbol{P}_I = \boldsymbol{p}(\boldsymbol{x}_I)$, $\boldsymbol{A} = \sum_{j \in V(\boldsymbol{x}_I^0)} W_{Ii} \boldsymbol{P}_j \boldsymbol{P}_j^T$ and with $W_{Ii} = W(\left\| \boldsymbol{x}_I^0 - \boldsymbol{x}_j^0 \right\|, h)$ is a monotonic and decreasing weight function on a distance threshold of h, beyond which it becomes null. In this work, we used the quartic spline weight function described in [HWJM10] where h is found by taking the mean distance of the k nearest neighbors of a position $\boldsymbol{x}$ and multiplying it by a small dilatation factor.

3.4 The corotational nodal elastic forces

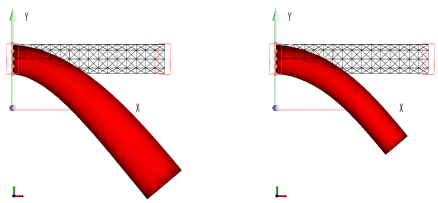
Similar to FE methods, and because the continuous integration terms in equation 5 are discretized into a sum over the integration points (equation 7), the continuous system is now reduced to a set of smaller systems of equations to be solved around each neighborhood. Before we describe the process of solving for the unknown displacements $\boldsymbol{u}$, let us start by assuming that they are already known. To derive the elastic forces of equation 4 applied to a particle i in the neighborhood of an integration point I, we look at the rate of change of internal elastic energy in the direction of $\boldsymbol{u}_i$, yielding:

$$\begin{aligned} \boldsymbol{f}_{I \to i}^{\text{elastic}} &= v_I \boldsymbol{\sigma}(\boldsymbol{u}_I) \cdot \delta \boldsymbol{\varepsilon} \\ &= v_I [\lambda (\nabla \cdot \boldsymbol{u}_I) \boldsymbol{I} + 2\mu \varepsilon(\boldsymbol{u}_I)] \cdot \delta \boldsymbol{\varepsilon} \\ &= v_I \boldsymbol{B}_i^T \boldsymbol{C} \sum_{j \in V(I)} [\boldsymbol{B}_j\ \boldsymbol{u}_j] \end{aligned} \quad (9)$$

where $\boldsymbol{C}_{ijkl} = \lambda \boldsymbol{I}_{ij} \boldsymbol{I}_{kl} + \mu (\boldsymbol{I}_{ik} \boldsymbol{I}_{jl} + \boldsymbol{I}_{il} \boldsymbol{I}_{jk})$ is the elasticity tensor, often reduced to a 6x6 matrix, and where $\boldsymbol{B}_i$ is the strain matrix and is defined as

$$\boldsymbol{B}_i^T = \begin{bmatrix} \phi_{i,x} & 0 & 0 & \phi_{i,y} & 0 & \phi_{i,z} \\ 0 & \phi_{i,y} & 0 & \phi_{i,x} & \phi_{i,z} & 0 \\ 0 & 0 & \phi_{i,z} & 0 & \phi_{i,y} & \phi_{i,x} \end{bmatrix} \quad (10)$$

Normally, a rigid transformation (translation or rotation) of the simulated body should not generate any elastic force since there is no deformation involved. Unfortunately, since both the strain tensor and constitutive model are linear approximations, rotations, which are non-linear transformations, will wrongly generate internal forces, often called "ghost forces" (see figure 2a).



(a) Ghost forces prevent the cylinder to rotate down.

(b) The corotational approach alleviates the effects of ghost forces by removing rotations before the computation of elastic forces.

Figure 2: Simulation of a cylinder fixed at the left and deformed by gravity (downward along the Y axis).

In order to minimize these ghost forces, we extend the work of [NPF05] and [BIT09] by introducing a corotational approach to our method. Whereas in FEM the rotation is extracted from an element, and in the SPH formulation of [BIT09] the forces are gathered around the particles, our method relies on the integration point neighborhood. Since these quadrature positions are not represented by unknown variables, their position does not get updated throughout the simulation. To solve this, we manually update their positions at the beginning of each time step by using the displacement of neighboring particles (equation 6). Using these updated positions, we construct a transformation matrix of each integration point subspace:

$$\boldsymbol{A}_I = \sum_{j \in V(\boldsymbol{x}_I^0)} v_j (\boldsymbol{x}_j - \boldsymbol{x}_I)(\boldsymbol{x}_j^0 - \boldsymbol{x}_I^0)^T \quad (11)$$

where v_j is the volume of a particle and is obtained by splitting the volume of the integration points evenly among its neighbors. We then use the stable SVD decomposition method of [PTVF07] to extract a rotation matrix $\boldsymbol{R}_I$ from the transformation matrix $\boldsymbol{A}_I$. Finally, we cancel this rotation from the displacement approximation in equation 9 to get the corotational forces:

$$\boldsymbol{f}_{I \to i}^{\text{elastic}} = v_I \boldsymbol{R}_I \boldsymbol{B}_i^T \boldsymbol{C} \sum_{j \in V(I)} [\boldsymbol{B}_j\ (\boldsymbol{R}_I^T \boldsymbol{x}_j - \boldsymbol{x}_j^0)] \quad (12)$$

The benefits of this approach are shown in figure 2b where ghost forces no longer appear.

3.5 Solving the static system

In the previous section, we described how to compute the localized elastic force $\boldsymbol{f}_{I \to i}^{\text{elastic}}$ applied to a given particle i in the vicinity of integration point I. The total elastic force on i is found by accumulating the contribution of every integration point. This elastic force definition is only useful when the current displacements $\boldsymbol{u}_j$

are known. Typically, in a static scheme (where there are no time dependent terms), the displacements are unknown, which implies that a linear system of equations must be solved in order to find them. When linear strain and stress tensors are used, displacements $\boldsymbol{u}$ can be factored to obtain the system $\boldsymbol{K}\ \boldsymbol{u} = \boldsymbol{f}^{\text{ext}}$ where $\boldsymbol{K}$ is the stiffness matrix and is constant in time. However, since our method relies on displacements relative to updated integration point positions due to our corotated approach, the stiffness matrix is no longer constant. Therefore, solving for the unknown displacements is done using an iterative Newton-Raphson method. Starting from an initial displacement, $\boldsymbol{u}^0$, we try through an iterative process to find a correction $\boldsymbol{\delta}_u$ that balances the linearized set of equations after n iterations:

$$\dot{\boldsymbol{K}}^{n-1}\ \boldsymbol{\delta}_u^n = \boldsymbol{f}^{\text{elastic}}(\boldsymbol{u}^0 + \boldsymbol{\delta}_u^{n-1}) + \boldsymbol{f}^{\text{ext}} \tag{13}$$

where $\dot{\boldsymbol{K}}$ is the tangent stiffness matrix obtained by deriving the force of equation 12 with respect to the displacement of the neighbors. The 3x3 sub-matrix $\dot{\boldsymbol{K}}_{ij}$ can be viewed as the linear action of the displacement $\boldsymbol{u}_j$ on the particle i:

$$\dot{\boldsymbol{K}}_{ij} = \sum_I v_I \boldsymbol{R}_I \boldsymbol{B}_i^T \boldsymbol{C} \boldsymbol{B}_j^T \boldsymbol{R}_I^T \tag{14}$$

where I represents an integration point influencing both particles i and j.

3.6 Solving the dynamic system

For dynamic schemes, time-dependent terms must be incorporated into the equations. For example, the gravity force involves the acceleration of particles, the damping force involves their velocity and collision forces involve the current state of the surfaces. To solve this time-dependent system, we decided to follow the Euler implicit time integration method [BW98]. Using the discrete formulation $\boldsymbol{M}\boldsymbol{a} - \boldsymbol{f}^{\text{elastic}} = \boldsymbol{f}^{\text{ext}}$, we derive the acceleration $\boldsymbol{a}$ and velocity $\boldsymbol{v}$ from the following equations:

$$\begin{aligned} \boldsymbol{M}\boldsymbol{a}^{t+\Delta t} &= \boldsymbol{f}^{\text{elastic}}(\boldsymbol{x}^{t+\Delta t}) + \boldsymbol{f}^{\text{ext}}(\boldsymbol{x}^{t+\Delta t}) + \boldsymbol{D}\boldsymbol{v}^{t+\Delta t} \\ \boldsymbol{v}^{t+\Delta t} &= \boldsymbol{v}^t + \Delta t \boldsymbol{a}^{t+\Delta t} \\ \boldsymbol{x}^{t+\Delta t} &= \boldsymbol{x}^t + \Delta t \boldsymbol{v}^{t+\Delta t} \end{aligned} \tag{15}$$

where $\boldsymbol{M}$ is a diagonal lumped mass matrix of the particles and $\boldsymbol{D}$ is a constant Rayleigh damping matrix. Since the forces at time $t + \Delta t$ are unknown, the following linear equation is obtained from a first order Taylor approximation :

$$(\boldsymbol{M} - \Delta t \boldsymbol{D} - \Delta t^2 \dot{\boldsymbol{K}})\boldsymbol{a}^{t+\Delta t} = \Delta t(\boldsymbol{f}^{\text{elastic}}(\boldsymbol{x}^t) + \boldsymbol{f}^{\text{ext}}(\boldsymbol{x}^t)) + \Delta t^2 \dot{\boldsymbol{K}} \boldsymbol{v}^t \tag{16}$$

This equation is finally solved using the iterative conjugate gradient method.

3.7 Surface displacement and external force mapping

So far our attention has been confined to the interior of the deformable body. The last and final step consists of extending the displacements derived in the previous steps to the object's surface using the shape functions. Since the surface of an object is usually represented by a mesh of triangles or quads, the displacement of these polygon vertices can be derived by evaluating the shape function of their neighboring particles (see figure 3b).

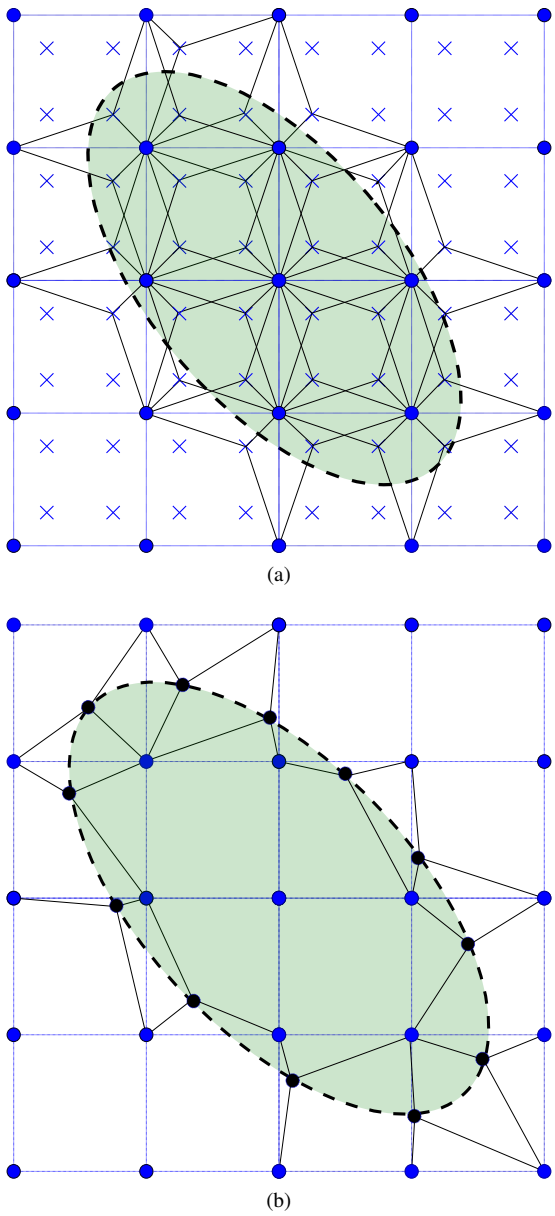


Figure 3: The similarities between (a) the relation between particles and integration points, and (b) the mapping of particles and the surface tesselation. Here, the green oval shape is the simulated object, the blue nodes represent the unknown degrees of freedom, the black nodes are the surface vertices and the blue crosses are the integration points.

While this method is trivial and very fast, it should be noted that other methods based on an implicit representation (see [MKN*04]) can also be used to produce even more visually satisfying results. However, given the objectives set earlier in this paper, we consider the displacement approach to be sufficient.

For the external forces applied to the surface mesh (such as collision and external pressure), the surface nodal forces are applied to the neighboring particles following the same general idea. Thus, for an external force $\boldsymbol{f}_s$ applied to a surface vertex at the initial position $\boldsymbol{x}_s^0$, the mapped force $\boldsymbol{f}_i^{\text{ext}}$ at a neighboring particle i becomes

$$\boldsymbol{f}_i^{\text{ext}} = \phi_i(\boldsymbol{x}_s^0)\boldsymbol{f}_s \tag{17}$$

4 RESULTS

In this section, we seek to demonstrate how our method positions itself with respect to the four criteria pursued in this work. Beginning with bending and stretching scenarios, the solutions are validated through convergence and precision analyses. Next, we outline the stability and simplicity of the method using various experiments involving multiple objects and materials. The computation times are given and were measured on an Intel(R) Core(TM) i7-6700K CPU @ 4.00 GHz computer with 16 GB of memory. No multithreading maneuvers were used, hence leaving place for future speed improvements. Our method was implemented as a plugin for the multi-physics open source SOFA Framework [FDD*12].

4.1 Convergence analysis

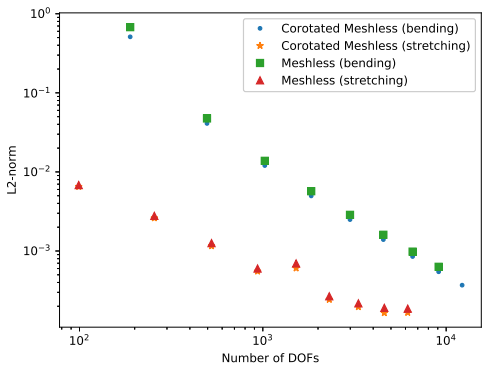
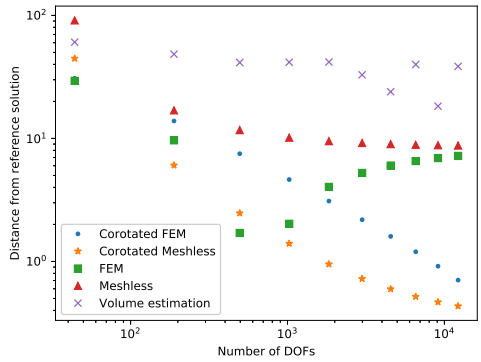
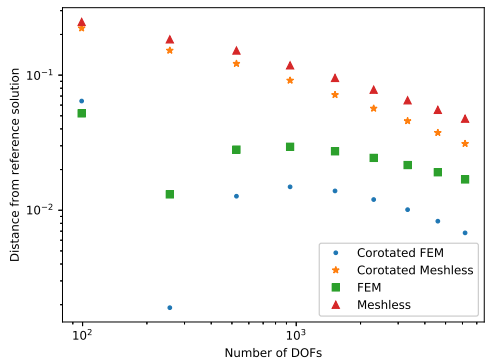


Figure 4: The convergence rate of our method in different scenarios. The error norm is obtained by comparing the n DOFs solution against the $(n-1)$ DOFs solution of the same variant of the method in the same scenario. The straight curve indicates a constant rate of convergence.

To verify that the implementation of our method works adequately from a numerical standpoint and that it results in accurate deformations, we performed



(a)



(b)

Figure 5: The accuracy of the solutions for the (a) bending and (b) stretching scenarios against a corotated FEM reference solution of 69k regular hexahedral elements.

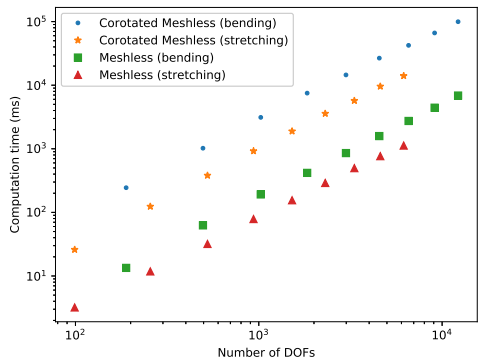


Figure 6: The computational times of the experiments

a convergence analysis using a FE solution as a reference. The first scenario was a regular beam of size $20x20x200\ mm^3$ placed horizontally, fixed at one end and subjected to a downward pressure force of $12\ N/mm^2$ at the other. The material used a Young modulus value of $50\ N/mm^2$ and a Poisson ratio of 0.45.

From this experiment, we gathered two measurements. The first one, presented in figure 4, shows the convergence rate of our method. It shows that, as the number of DOFs increases, the static solution tends towards a unique solution at a constant rate, whether that solution is accurate or not. The second one, presented in figure 5a, validates the accuracy of the converged solution against a reference solution. In this work, we used a corotated FEM solution of nearly 70k regular hexahedral elements as a reference.

To demonstrate issues with estimating the volume of an object without a background grid, we also simulated the beam with a nodal integration method where node mass and volume were estimated by sampling neighboring nodes as in [MKN*04]. While the estimated density is very accurate, the corresponding mass and volume are much higher than the actual ones and result in higher stiffness of the simulated object. Furthermore, the estimation is affected by particle distribution and will thus vary as the number of DOFs changes. This means that the simulation will not converge to a specific solution as we increase the number of DOFs.

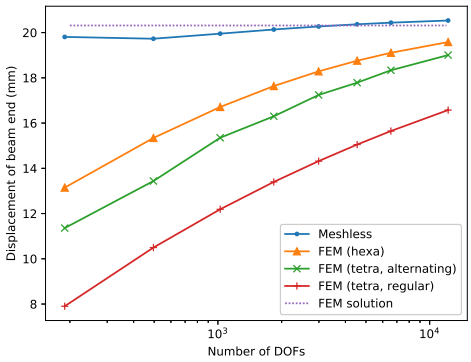


Figure 7: The "locking" artifact of linear tetrahedrons when used on an almost incompressible bending beam. The distances of displacement from the tip of the solutions are shown.

The bending of a beam is also a good way to illustrate one of the inherent meshing problems that come with FE methods. As those methods need to generate a mesh that conforms to the boundary surface, tetrahedron meshes are generally preferred over hexahedrons since they are well suited for automatic meshing methods. However, the solutions of linear tetrahedron-based FEM can vary a lot: the under-integrating aspect during the computation of the stiffness matrix can introduce large numerical errors, also known as the "locking" effect. By building two tetrahedron FE models from the regular hexahedron one, where both have 6 tetrahedrons per cube, but with different orientations, we can clearly see how this numerical error impacts the solution. Both have exactly the same number of elements, system unknowns and element shapes. However, as shown in figure 7, they converge to different solutions. In both cases, these tetrahedrons would have been accepted as "well-shaped elements" by most automatic meshing software. We can also see how our method and the hexahedron one does not suffer from this numerical constraint. Conversely, while the hexahedron FE method is trivial to implement for a squared cross-section beam, it remains very hard to apply to general complex objects. This demonstrates an attractive benefit of our methods over traditional FE methods.

In figure 4 and 5b, the experiment is repeated but this time through a stretching scenario. Here, using the same previously used material, a pressure force exerting 1500 N/mm^2 was used in a direction parallel to the beam. The computation times from both bending and stretching scenarios are outlined in figure 6. To solve the static systems, we used the Newton-Raphson method described in equation 13 with a maximum of 100 iterations and a residual threshold of 0.00001.

4.2 Multiple materials

To illustrate the process of discretizing an object with different material properties, figure 8 shows a cylinder fixed at its center and deformed by gravity. Here, two background grids of different material properties were placed side by side, illustrating the simplicity of setting the different material properties.

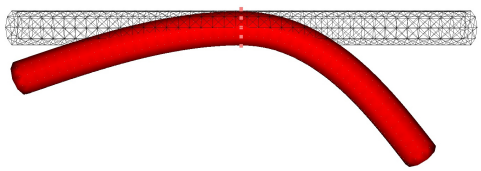
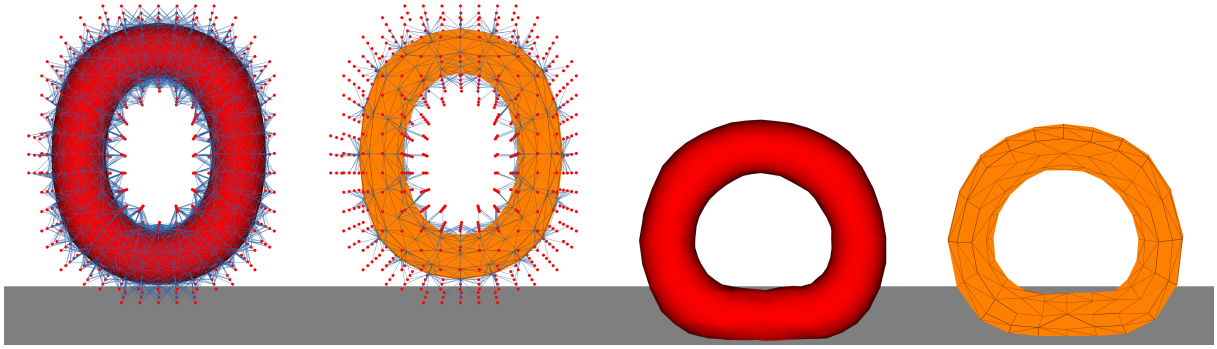


Figure 8: Material properties can vary inside a single object. The cylinder is fixed at its center and gravity is applied. The left half of the cylinder has a much higher Young's modulus than the right half.

We can imagine how this could be extended to complex objects where some parts must be stiffer or heavier than the others. A gradient of the material properties could also be added around the boundary regions of the different parts, smoothing out the change in material. The inherent simplicity of dealing with a cloud of particles then allows for a lot of flexibility to the user, either for determining how the objects should behave or to improve the accuracy of the solution in some specific regions of the object.

4.3 Surface mapping

The result of mapping the surface representation onto the degrees of freedom is best demonstrated visually, in figure 9. This figure shows the various object representations that are manipulated during a simulation. The master state, represented by the degrees of freedom (red



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

Figure 9: Mapping between a surface and the deformation state of a torus. (a-b) Relation between the master state (red circles) and its slave surfaces: the visual (red) and contact (yellow) tesselations. (c-d) Resulting mapped surfaces after the contact with the floor.

circles in figures 9a and 9b) describes the deformation of the object, while visual and contact model representations completely depend on that state to get their final shapes (figures 9c and 9d). The contact forces can be obtained through different methods. In this work, a simple penalty based contact force was sufficient for our demonstration purposes, even when used with collisions between deformable objects as shown in figure 10.

5 CONCLUSION

We have presented a method for animating deformable objects at interative framerates that require no polyhedral meshing of the volume. Where finite element methods require well-formed and well-placed elements inside their domain, our method only needs a cloud of points to solve the system of unknown displacements. Unlike other meshless methods based on nodal density integration, we use a regular background grid which does not need to conform to the surface mesh to efficiently integrate the Galerkin formulation of elasticity PDEs to be solved, hence improving the accuracy of the solution. To minimize numerical errors and by using the integration points as a local reference frame, a rotation matrix was extracted from the neighborhood of those frames and used to reduce the effects of large nonlinear displacements. We have shown that this method is both stable and accurate by presenting convergence and precision analyses. We have also presented images directly extracted from various simulations involving collisions. These time dependent simulations followed an implicit time integration scheme that is well suited for interactive applications incorporating large and possibly inconsistent steps.

While this method is promising, there is however room for improvement. Since the computation involves neighborhoods containing more nodes than its FE method counterparts, the resulting computation time is a little bit heavier. Conversely, this can be greatly improved by using multi-core computers and exploiting the symmetry in elementary stiffness matrices by node numbering optimizations. The benefits of this method could also be explored in topological change scenarios, such as cutting and plastic deformations. Finally, we plan on doing extended convergence analyses to establish optimal neighborhood configuration based on the sparsity of the nodes and integration points to further improve convergence rates.

Acknowledgements. This study was supported by H2020-MSCA-ITN Marie Skłodowska-Curie Actions, Innovative Training Networks (ITN) - H2020 MSCA ITN 2016 GA EU project number 722068 High Performance Soft Tissue Navigation (HiPerNav).

6 REFERENCES

[BC96] Bro-Nielsen, Morten and Cotin, Stéphane. "Real-time volumetric deformable models for surgery simulation using finite elements and condensation". *Computer Graphics Forum* 15 (1996), 57–66.

[BIT09] Becker, Markus, Ihmsen, Markus, and Teschner, Matthias. "Corotated sph for deformable solids". *Natural Phenomena*. 2009, 27–34.

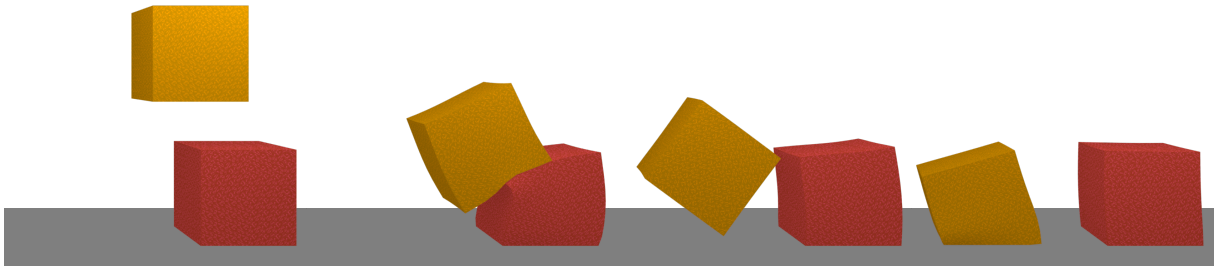
[BKO*96] Belytschko, T, Krongauz, Y, Organ, D, et al. "Meshless Methods: An Overview and Recent Developments". *Computer Method in Applied Mechanics and Engineering* 139 (1996), 3–47.

[BLG94] Belytschko, Ted, Lu, Yun Yun, and Gu, Lei. "Element-free Galerkin methods". *International Journal for Numerical Methods in Engineering* 37.2 (1994), 229–256.

[BW98] Baraff, David and Witkin, Andrew. "Large steps in cloth simulation". *Proceedings of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH '98*. 1998, 43–54.

[CDA99] Cotin, Stéphane, Delingette, Hervé, and Ayache, Nicholas. "Real-time elastic deformations of soft tissues for surgery simulation". *IEEE Transactions on Visualization and Computer Graphics* 5.1 (1999), 62–73.

[DLLW16] Dong, Yi, Liu, Xuemei, Li, Hairui, and Wang, Zhenkuan. "A Nonlinear Viscoelastic Meshless Model for Soft Tissue Deformation". *2016 International Conference on Virtual Reality and Visualization (ICVRV)* (2016), 204–211.



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

Figure 10: Collision between two deformable objects. Both cubes are deformed while colliding with each other.

[DRTZ16] Dehghan, Mohammad Reza, Rahimi, Abdolreza, Talebi, Heidar Ali, and Zareinejad, Mohammad. "A three-dimensional large deformation model for soft tissue using meshless method". *International Journal of Medical Robotics and Computer Assisted Surgery* 12.2 (2016).

[FDD*12] Faure, François, Duriez, Christian, Delingette, Hervé, et al. *SOFA; a Multi-Model Framework for Interactive Physical Simulation*. Vol. 11. 2012, 283–321.

[Fel00] Felippa, Carlos A. *A systematic approach to the element-independent corotational dynamics of finite elements*. Tech. rep. Technical Report CU-CAS-00-03, Center for Aerospace Structures, 2000.

[HWJM10] Horton, Ashley, Wittek, Adam, Joldes, Grand Roman, and Miller, Karol. "A meshless Total Lagrangian explicit dynamics algorithm for surgical simulation". *International Journal for Numerical Methods in Biomedical Engineering* 26.8 (2010), 977–998.

[MHJW12] Miller, K., Horton, A., Joldes, G. R., and Wittek, A. "Beyond finite elements: A comprehensive, patient-specific neurosurgical simulation utilizing a meshless method". *Journal of Biomechanics* 45.15 (2012), 2698–2701.

[MHTG05] Müller, Matthias, Heidelberger, Bruno, Teschner, Matthias, and Gross, Markus. "Meshless deformations based on shape matching". *ACM transactions on graphics (TOG)*. Vol. 24. 3. ACM. 2005, 471–478.

[MKN*04] Müller, Matthias, Keiser, Richard, Nealen, Andrew, et al. "Point based animation of elastic, plastic and melting objects". *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association. 2004, 141–151.

[NMK*06] Nealen, Andrew, Müller, Matthias, Keiser, Richard, et al. "Physically based deformable models in computer graphics". *Computer graphics forum*. Vol. 25. 4. Wiley Online Library. 2006, 809–836.

[NPF05] Nesme, Matthieu, Payan, Yohan, and Faure, François. "Efficient, Physically Plausible Finite Elements". *Eurographics* (2005), 77–80.

[PGBT18] Peer, Andreas, Gissler, Christoph, Band, Stefan, and Teschner, Matthias. "An implicit SPH formulation for incompressible linearly elastic solids". *Computer Graphics Forum*. Vol. 37. 6. Wiley Online Library. 2018, 135–148.

[PTVF07] Press, William H, Teukolsky, Saul a, Vetterling, William T, and Flannery, Brian P. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. 2007, 1262.

[SCL*04] Sorkine, Olga, Cohen-Or, Daniel, Lipman, Yaron, et al. "Laplacian surface editing". *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. ACM. 2004, 175–184.

[SSP07] Solenthaler, Barbara, Schläfli, Jürg, and Pajarola, Renato. "A unified particle model for fluid-solid interactions". *Computer Animation and Virtual Worlds*. Vol. 18. 1. 2007, 69–82.

[WGJ*16] Wittek, Adam, Grosland, Nicole M., Joldes, Grand Roman, et al. "From Finite Element Meshes to Clouds of Points: A Review of Methods for Generation of Computational Biomechanics Models for Patient-Specific Applications". *Annals of Biomedical Engineering* 44.1 (2016), 3–15.

[ZWJ*14] Zhang, G. Y., Wittek, A., Joldes, G. R., et al. "A three-dimensional nonlinear meshfree algorithm for simulating mechanical responses of soft tissue". *Engineering Analysis with Boundary Elements* 42 (2014), 60–66.