Fast Meshless Simulation of Anisotropic Tearing in Elastic Solids

by

Omar Hesham

B. Information Technology, Carleton University, 2009

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs

in partial fulfillment of the requirements for the degree of

Master of Science

in

Information and Systems Science (Systems Engineering)

Department of Systems and Computer Engineering

Carleton University

Ottawa, Ontario, Canada

November 2011

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-87835-4

Our file Notre référence
ISBN: 978-0-494-87835-4

Abstract

Dynamic models to simulate tearing of soft elastic bodies are an essential element of various medical and surgical training simulators. These models are also finding increased use in film and gaming applications, where control over the quality and style of the final output is highly valued. There is a general lack of models specifically designed to control tearing patterns, and in this thesis, we present our effort towards a soft-body tearing method that provides simple parametrizations to control the fracture independently from the elastic properties of the soft body simulation. Our parameters can influence how clean-cut or feathered the tear is, in addition to allowing anisotropic influence using an embedded fibre model in the elastic body. We also aim for a real-time implementation suitable for interactive environments, and our meshless solution is discussed in a context that is aware of the importance of unified physics solvers.

Acknowledgments

This thesis would not have been possible without the support and advice from my ever patient, trusting and resourceful supervisor, Professor Chris Joslin. I also want to thank Meagan Leflar for her help with coding sample implementations and Chris Taylor for all the academic discussions and the pleasant company in the labs.

I'm also thankful for my loving family, who supported me throughout my university experience.

Parts of this research have been funded by an Ontario Graduate Scholarship.

Table of Contents

List of Tables

List of Figures

List of Acronyms

Acronyms	Definition
PDE	partial differential equation
ODE	ordinary differential equation
SPH	smoothed particle hydrodynamics
MSS	mass-spring system
MLS	moving least squares
FEM	finite element method
BEM	boundary element method

Chapter 1

Introduction

Computer models to solve deformable body problems and fracturing simulations were developed originally to simulate solutions for the engineering and applied sciences fields. These methods soon found their way into the computer graphics field, applying their strengths to simulate various natural dynamic phenomenon like cloth, fluids and soft bodies. Research that is specific to computer graphics models is geared towards stability and visual realism, as opposed to absolute theoretical accuracy. Speed and efficiency are also valued greatly when comparing methods against each other. These developments find their primary application in film, game and medical simulation, and this thesis is no exception when it comes to prioritising those properties. We present a method to simulate fast tearing in elastic bodies that gives the user control over the fracture style and direction, while aiming at a real-time implementation for interactive environments.

Automated computer models are designed to aid the artist or the developer in efficiently simulating a physical behaviour, but in some cases, they end up being restricted to what the results of the simulation are. This is why control over the dynamic simulation is important, and not just in film and animation where the value of control is obvious, but also in medical surgery simulators. Ideally, an accurate model would be able to simulate surgical procedures for training new surgeons. However,

research shows [1,2] that the realism from the perspective of an experienced physician is valued higher than theoretical accuracy, raising the need for control over these interactive simulations. We take this into consideration when designing our flexible tearing framework.

While soft and viscoelastic body simulation can be done efficiently for highly dynamic real-time gaming environments, fracturing or tearing those objects apart remains a challenge at those frame rates. This drove studios to preprocess their models by breaking them apart and holding those pieces together with artificial cohesive "glue" values. While this can give visually appealing and convincing results for breakable rigid bodies, it is very limiting for soft bodies that can tear at arbitrary locations.

This motivated us to formulate a soft body tearing framework that can run in interactive environments and allow independent high level control over the fracturing style, while interacting elegantly with the physical model used to simulate the body's elastic behaviour.

1.1 Goals and Contributions

Tearing is defined as the breaking apart of a deformable object as a result of external or internal forces, but without the use of a cutting instrument [3]. This thesis presents our research efforts towards an accurate tearable soft-body simulation with an emphasis on the following properties:

- A tearing method that provides stylistic and directional control with simple, easily implementable parameters that are independent from but work together with the underlying material properties used for the elastic simulation.

- Have the potential for a real-time implementation without majorly compromising the accuracy of the simulation.
- Be based on a dynamics framework that can be embedded or extended into a unified physics solver (easy transition, extension and coupling with other physical systems like fluid and cloth dynamics.)

We have successfully met these goals using our implementation, through which we contributed the following:

- Extending Becker et al.'s Smoothed Particle Hydrodynamics (SPH) elastic simulation framework [4] to support topological changes through tearing (or fracture, terms used interchangeably). Their meshless framework was chosen for its realistic simulation of elasticity at *real-time* rates while also being extensible to fluid and rigid simulation, as previously demonstrated by Solenthaler el al. [5].
- Introduced the concept of a *fracture disk* that is inserted at any point of fracture throughout the simulation domain and, with the aid of an empirically developed transparency map, affords the end user a degree of control over the style of the tear (sharp and clean-cut vs. soft and feathered), decoupled from the elastic properties of the soft body being torn.
- To control the tearing direction, we presented an improved anisotropic tearing model that uses a vector field, called *fibres*, to influence the orientation of the tear in an isotropic material. We introduce the use of strain-rate to modulate this influence and to avoid a completely geometric and artificial tearing pattern. We also describe a fast and simple method of maintaining the fibre field definition during the simulation, while robustly handling situations of fracture and topological changes in the material.

1.2 Thesis Organisation

This thesis is organised into five chapters. Chapter 1 contains an introductory look at the goals of the thesis and a summary of our contributions. The relevant existing methods are discussed in Chapter 2 with an emphasis on frameworks that implement fracturing and controlled tearing, followed by an overview of continuum mechanics basics and a detailed look at Becker et al.'s real-time elasticity framework. Our tearing method is detailed in Chapter 3, followed by a discussion of the results in Chapter 4, before finally concluding in Chapter 5 with an overview of our contributions and the future potential of progress stemming out of this research.

Chapter 2

Background

2.1 Review of Existing Methods for Meshless Elasticity and Tearing Simulation

Ever since Terzopoulos et al.'s [6] seminal work on the dynamic simulation of deformable bodies and their fracture, this area of computer graphics research has seen a rapid increase in development, focusing on stable, visually realistic and efficient methodologies for simulating various natural physical phenomenon. Specific to fracture simulation, earlier methods heavily featured the Finite Element Method (FEM) discretization, thanks to its connected mesh definition and the ease of disconnecting them wherever fracture events occurred. FEM also provided the desirable property of convergence (as the resolution of the mesh increases, the method approaches the true continuum solution) and allowed easy textured surface embedding by simply treating the boundary faces of the FEM mesh as the visual surface to render. Examples of efficient FEM fracture include O'brien et al.'s [7] and its extension to ductile fracture [8], and Irving et al.'s [9] very stable formulation for large and even inverted mesh deformations. FEM found further practical use in real-time gaming environments [10] [11] and a whole host of specialised medical and surgical scenarios [1].

However, remeshing around topological changes while avoiding the generation of irregularly shaped and problematic elements remains a non-trivial challenge for FEM methods. This is important in tearing simulation, considering the need for fracture lines that cut through the mesh at arbitrary locations. Recent attempts to remedy this problem [12] [13] are still far from any real-time implementation, making them non-viable for consistently interactive high quality environments.

Meshless methods have recently attracted attention for their flexibility and resampling freedom. In essence, the domain is discretized into disconnected particles (nodes) where the dynamic properties are evaluated, and each has an influence on the rest of the continuous body that decreases as we go further away from it. They do not suffer from the remeshing problems of FEM methods and have been found particularly useful for fracture simulation. We continue this section with a review of the meshless methods most relevant to our research and that have an emphasis on tearable bodies.

Muller et al. [14] used moving least squares (MLS) to calculate a first-order-accurate estimation of the deformation gradient and successfully showcased the versatility of meshless methods in handling elastic, plastic and viscoelastic simulation seamlessly in the same framework and at interactive rates. This was further improved upon by Gerszewski et al. [15], eliminating the need for an initial reference position and relying instead on the velocity gradient for stress computation, albeit at a higher computational cost. The major problem with MLS-based methods is their moment matrix inversion step which degenerates when the neighbourhood of a particle contains less than three particles or is in a collinear or coplanar configuration. This meant that the method could not be used to simulate 1D (e.g. ropes) or 2D (e.g. thin membrane) objects. The topology in these approaches does not change throughout the simulation, except implicitly when particles leave or enter each others neighbourhood radii.

Smoothed particle hydrodynamics (SPH) is a meshless method, originally developed a few decades ago to model cosmo-galactic events [16], that later became very popular in computer graphics for real-time Lagrangian fluid simulation due to its ability to handle rapidly changing neighbourhoods with high temporal coherency and stability [17] [18] [19]. Solenthaler et al. [5] devised an SPH framework to model a relatively large variety of physical phenomenon including rigid, deformable and fluid objects. In addition, they were able to incorporate existing SPH models of thermal flow and use it to determine the material state of the object with smooth transition periods. This enabled them to simulate melting and freezing, in a highly controllable configuration. However their elasticity formulation suffered from being rotationally variant, causing incorrect handling of rotating elastic bodies that relied on an initial reference position. This was later corrected by Becker et al. [4] in their corotated SPH formulation for elastic bodies.

Pauly et al. [20] introduced fracture into the MLS framework using a visibility criteria between particles to update neighbourhood information for MLS computation and explicitly change object topology. They tracked each crack front explicitly after its initiation at a highly stressed area then formed new surface panels using the crack lines. The strength of their method lies in their meshless resampling schemes which provide higher accuracy around surfaces and fracture lines. However, their crack propagation happens only in a radial fashion, does not support coplanar and collinear particle configurations, and does not run in real-time.

Liu et al. [21] presented a meshless method to simulate fracture in brittle solids. They used rigid dynamics for regular simulation and the Meshless Local Petric-Galerkin (MLPG) method to evaluate stresses throughout the body during a fracturing event. Their quasi-static MLPG formulation, where inertia is not taken into consideration, tackles the difficult problem of propagating brittle cracks throughout a relatively large span of the body over a small period of time without having a

large impact on the simulation running time. Unlike in Pauly et al.'s [20], they do not explicitly track and propagate individual crack fronts. Instead, they use a damage-based criteria along with a clustering method (similar to k-means and bubble clustering used in preprocessing graphs for optimal parallel computing [22]) to partition the object and apply rigid simulation to each partition independently. User control over the fracture pattern was allowed through a custom weighted vector field to guide the clustering part of their algorithm. While this approach is fast and works well for brittle fracture in rigid solids, it also forces the creation of a new partition at every fracturing event, making it unsuitable for our elastic soft body tearing where partial cuts are allowed.

One of the few real-time medical simulators [23,24] that provide anisotropic tearing is Allard et al.'s cataract surgery simulator [25]. In their implementation, they use a 2D FEM model along with a vector field (they called *fibre field*) to define the desired direction of tear propagation. In contrast to the previously discussed methods, however, their approach is purely physical, relying on a complete anisotropic elasticity simulation, which in turn directly results in anisotropic tearing simulation. While their results are convincing, their method does not allow for anisotropic fracture control in isotropic materials.

Our controlled tearing method is highly influenced by the concepts discussed in these previous three papers, [20] [21] [25], and adapts their advantages to meet our own controllable soft-body tearing goals.

2.2 Continum Mechanics for Elastoplastic Simulation

In this section, we present a quick overview of soft-body simulation basics and the underlying continuum mechanics. For a complete review we highly recommend the extensive survey in [26], in addition to [27] which covers these topics in detail and with rigour, and [28] for specific real-time efficient implementations. In this thesis, we cover concepts most relevant to our simulation.

2.2.1 Elasticity Theory

A given elastic body is defined using its original reference position $\mathbf{x}^0$ vector field and its currently deformed position is denoted with the $\mathbf{x}$ vector field. The current displacement $\mathbf{u} = \mathbf{x} - \mathbf{x}^0$ represents the displacement vector field from the original shape. The variation in this displacement field $\nabla\mathbf{u}$ is then used to compute the amount of strain (deformation) ε in the body. Common strain tensors used in computer graphics include Green's non-linear tensor:

$$\varepsilon = \frac{\nabla\mathbf{u} + [\nabla\mathbf{u}]^T + [\nabla\mathbf{u}]^T\nabla\mathbf{u}}{2} \tag{2.1}$$

and Cauchy's linear tensor:

$$\varepsilon = \frac{\nabla\mathbf{u} + [\nabla\mathbf{u}]^T}{2} \tag{2.2}$$

Stress σ, measured in Pascals, is the resulting force per unit area created by this strain, and is also a 2nd order tensor itself. The strain-stress relationship is governed by the material's constitutive law:

$$\sigma = \mathbf{C}\varepsilon, \tag{2.3}$$

where $\mathbf{C}$ is the 6x6 elasticity matrix, defining the physical properties of the material like stiffness, volume conservation and isotropy. In computer graphics, $\mathbf{C}$ is usually simplified to be only dependant on the two Lamé constants: Poisson's ratio and Young's modulus, which are common descriptors of mechanical behaviour.

Continuum elasticity dynamics can be obtained by solving the following Partial Differential Equation (PDE):

$$\rho \mathbf{a} = \nabla \cdot \sigma + \mathbf{f}_{ext} \tag{2.4}$$

where ρ is material density, $\mathbf{f}_{ext}$ is the net external force (gravity, collisions, user interaction, etc.), and $\nabla \cdot \sigma$ represents the force vector created by internal deformation. As the equation is solved, we are able to find the acceleration $\mathbf{a}$, which enables us to advance the dynamic system in time.

Discretization

The PDE in equation (2.4) is just another form of Newton's second law of motion $\mathbf{f} = m\mathbf{a}$. However, in its current form, it needs to be solved over the entire continuum of the problem domain, making an analytical solution impractical and near impossible to efficiently compute. Instead, the spatial domain is discretized into a finite set of disjoint elements or nodes, and the PDE unknowns are numerically solved only at those nodes. The Finite Element Method (FEM) is one such solution that is grossly popular in the engineering and science fields, and has enjoyed decades of development and research to achieve a high degree of accuracy and efficiency that can be tuned and optimised for various PDE problems. In FEM, the volume (or area in 2D) of the problem domain is discretised into an irregular grid of connected elements, most commonly simplices (triangles in 2D, tetrahedra in 3D, etc.). For each element, a simplified version of the PDE in equation (2.4) is solved and weighted at every vertex

to compute an approximation of the true deformation of this element. Neighbouring elements in FEM are explicitly connected to each other, so the force computation at a given vertex, for example, will be the net sum of the deformation forces experienced in all the elements that share this vertex.

Another lesser-known approach to discretization relies on a finite set of particles to represent the problem domain. In contrast to FEM, these particles (or nodes, interchangeable terms) are not joined or connected, but rather interact freely with spatially close particles to solve for the required physical quantities. Hence the method being referred to as meshless. In the most common case, the particles are distributed evenly across the domain (not necessarily on a uniform grid), with each particle representing a localised volume on which a modified version of the PDE (2.4) is solved. Meshless techniques for computer graphics include Shape Matching [29] [30], Smoothed Particle Hydrodynamics [5], and Moving Least Squares [14] to name a few, and their relative aspects were reviewed in section 2.1.

For a dynamic simulation that involves tearing, the problem domain and its topology are expected to change with each tearing event. Figure 2.1 illustrates how FEM methods are prone to produce degenerate and ill-shaped elements (too long, too narrow, area nearing zero, not a simplex anymore, etc.) that result in inaccurate PDE calculations, instability, and visual artifacts. The typical solutions require some form of post-processing of the FEM mesh, in most cases local remeshing and realignment is performed, which is a non-trivial problem both in the geomteric sense and in terms of maintaining and redistributing the physical properties correctly onto the new FEM mesh. On the other hand, meshless methods do not have any connections to re-align, and instead automatically adjust their computations according to the new neighbourhood of adjacent particles. Thus, a meshless representation provides a suitable discretization for topologically changing dynamic simulations. This is also evident from the practicality of particle-based fluids simulation in real-time environments [17].

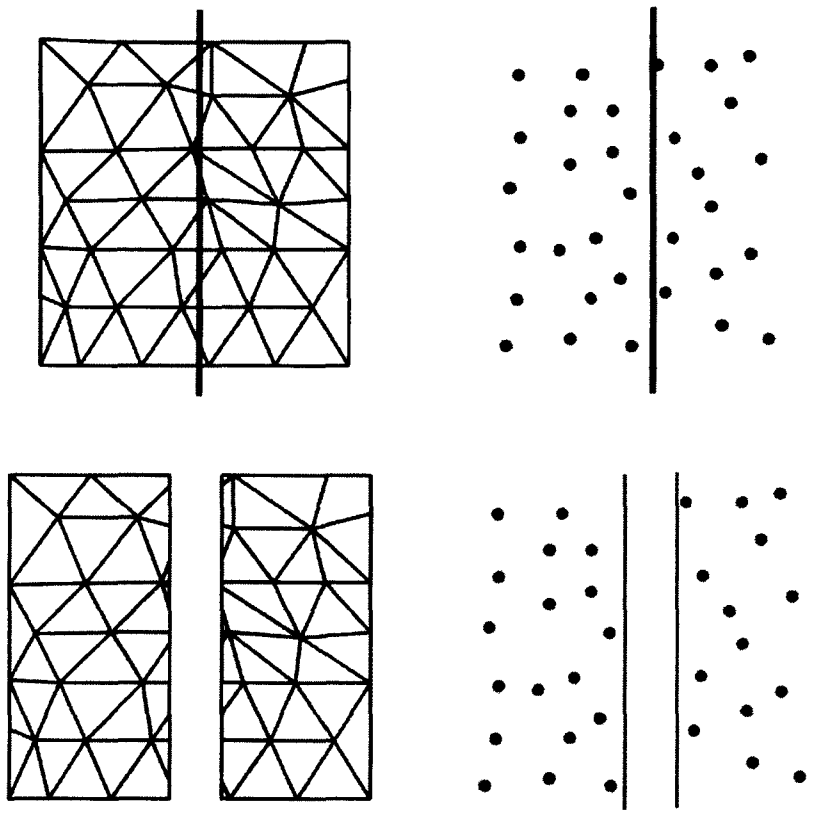

Figure 2.1: Tearing an FEM mesh (left) results in undesirable elements, highlighted here in orange, which need to be dealt with and remeshed before continuing the simulation. A particle-based meshless method (right) does not suffer from this problem, and instead relies on the new neighbourhood to automatically adjust its dynamic quantities.

This was a significant motivator for our choice of a meshless simulation.

Displacement Field Gradient

This is perhaps a good time to highlight why the displacement vector field *gradient* $\nabla \mathbf{u}$ is more important in soft body simulation than the displacement field $\mathbf{u}$ itself. Note that if the displacement field was constant we'd be witnessing a translation. In this case $\nabla \mathbf{u} = 0$, correctly reflecting the absence of deformation. It is also useful

to look at it from the perspective of stress and elastic force computation. Because continuum mechanics does not deal with concentric points of mass, it does not matter how much a given point has travelled away from its original orientation, rather we are interested in the deformation of the neighbourhood around that point. This further emphasizes the continuum nature of the formulation (and any discretized solution) and ensures stress is induced in the actually strained locations.

Linearity in Different Contexts

There are two main components to consider linearising in the elasticity formulation. The first is the strain tensor description which is geometric in nature. The non-linear strain tensor represents the deformation strain fairly accurately as opposed to its rotationally variant linearisation. Linear strain is faster to compute and integrate, but is only useful in cases with small deformations that do not involve large rotational deformations. The second component to consider is the stress-strain (constitutive) relationship which is physical in nature and depends entirely on the type of material being simulated. Hooke's law is linear and materials following this model are called Hookean materials. A non-linear strain coupled with a linear stress is sufficient for accurate simulation in most cases. Highly non-linear materials like rubber and soft biological tissue, however, will require a non-linear stress tensor for accurate representation, making them a challenge for real-time simulation.

2.2.2 Time Integration

In order to advance the dynamic system in time, an analytical solution is too complex to formulate and solve, so we instead use numerical methods. Newton's seoncd law of motion $\mathbf{f} = m\mathbf{a}$ is used to find the acceleration vector $\mathbf{a}$ for an element, given the total forces acting on it. Starting from the current time t and using a constant time

step Δt several options are available to integrate our system and find the new velocity $\mathbf{v}_{t+\Delta t}$ and position $\mathbf{x}_{t+\Delta t}$:

Explicit Euler

This is the simplest and fastest method where the quantities on the right hand side of the equations are known (explicit). It suffers from high instability, however, at large time-steps as it adds energy to the system and can overshoot the expected motion trajectory:

$$\begin{aligned} \mathbf{x}_{t+\Delta t} &= \mathbf{x}_t + \Delta t \mathbf{v}_t \\ \mathbf{v}_{t+\Delta t} &= \mathbf{v}_t + \Delta t \mathbf{a}_t \end{aligned} \tag{2.5}$$

Implicit Euler

A much more stable solution, involving unknowns on both sides of the equation. It is very stable and allows larger time steps, however at the high computational cost of solving an algebraic system of equations to find the new position and velocity. It also decreases energy in the system:

$$\begin{aligned} \mathbf{x}_{t+\Delta t} &= \mathbf{x}_t + \Delta t \mathbf{v}_{t+\Delta t} \\ \mathbf{v}_{t+\Delta t} &= \mathbf{v}_t + \Delta t \mathbf{a}_{t+\Delta t} \end{aligned} \tag{2.6}$$

Verlet Integration

A fast and stable alternative that skips velocity computation and directly obtains the new position. However, it requires two force computations per frame:

$$\mathbf{x}_{t+\Delta t} = 2\mathbf{x}_t - \mathbf{x}_{t-\Delta t} + \Delta t^2 \mathbf{a}_t \tag{2.7}$$

Symplectic Euler

The idea here is to compute velocity explicitly, while computing the position implicitly. This is as fast as Explicit Euler, but with similar stability to Verlet methods:

$$\begin{aligned} \mathbf{v}_{t+\Delta t} &= \mathbf{v}_t + \Delta t \mathbf{a}_t \\ \mathbf{x}_{t+\Delta t} &= \mathbf{x}_t + \Delta t \mathbf{v}_{t+\Delta t} \end{aligned} \tag{2.8}$$

With the new velocity $\mathbf{v}_{t+\Delta t}$ and position $\mathbf{x}_{t+\Delta t}$, we have reached the end of the simulation cycle for the current frame.

2.3 Meshless Simulation of Elastic Solids

In this section, we review the specifics of the corotated Smoothed Particle Hydrodynamics (SPH) elasticity method outlined by Becker et al. [4]. Their formulation is fast, rotationally invariant, supports 2D and 1D soft bodies, and can be extended to allow transitions between solids and fluids.

2.3.1 Meshless Representation

The object in 3D space is discretized into a set of n disconnected particles $P = \{p_0, p_1, \ldots, p_n\}$. Each particle i stores its initial position $\mathbf{x}_i^0$ representing the object under no deformation, and the current position in a given frame of animation as $\mathbf{x}_i$ with the displacement $\mathbf{u}_i = \mathbf{x}_i - \mathbf{x}_i^0$. It is preferable to distribute the particles as evenly as possible to ensure similar accuracy across the homogeneous soft body's domain. Following a lattice structure is acceptable. However, we prefer to distribute particles uniformly randomly followed by a just few iterations (2 to 5) of a Centroidal Voronoi relaxation [31].

We define the neighbourhood list of particle i as the set of all particles j within a given radius h. Neighbourhoods are defined in the initial configuration and this elastic structure is maintained throughout the simulation. The only neighbourhood update allowed is due to a fracturing event where the influence of j is either reduced or completely removed from i.

2.3.2 Estimating $\nabla \mathbf{u}$ using Smoothed Particle Hydrodynamics (SPH)

The original SPH formulation [16] leverages the known distribution of the particle set to derive a smooth approximation of a continuous attribute function $A(\mathbf{p}_i)$. To calculate the value of the function at any point $\mathbf{p}_i$, a finite set of neighbouring particles $\mathbf{p}_j$ with mass m_j and density ρ_j are sampled, weighted by a kernel $W(\mathbf{p}_i - \mathbf{p}_j, h)$ where h defines the influence (neighbourhood) radius. The approximated function $\tilde{A}(\mathbf{p}_i)$ is then:

$$\tilde{A}(\mathbf{p}_i) = \sum_j \frac{m_j}{\rho_j} A(\mathbf{p}_j) W(\|\mathbf{x}_i - \mathbf{x}_j\|, h). \tag{2.9}$$

Every particle is given a constant mass m_i. Density ρ_i at the particle can be computed as:

$$\rho_i = \sum_j m_j W(\|\mathbf{x}_i^0 - \mathbf{x}_j^0\|, h) \tag{2.10}$$

Volume $\tilde{v}_i$ represented by each particle is considered constant throughout a regular elastic simulation and is precomputed for each particle as:

$$\tilde{v}_i = \frac{m_j}{\rho_j} \tag{2.11}$$

Recall from section 2.2 that the gradient $\nabla \mathbf{u}$ of the displacement field is needed for the computation of strain, stress and the associated elastic forces. We start by

using the SPH formulation in (2.9) to approximate the smooth displacement field function $\mathbf{u}$ over our set of particles. At particle $\mathbf{p}_i$:

$$\mathbf{u}_i = \sum_j \tilde{v}_j \mathbf{u}_{ji} W(\|\mathbf{x}_i^0 - \mathbf{x}_j^0\|, h), \tag{2.12}$$

where $\mathbf{u}_{ji}$ is the vector representing the relative displacement, calculated as the difference between the displacements of neighbouring particles j and i as $\mathbf{u}_{ji} = \mathbf{u}_j - \mathbf{u}_i$.

A very useful property of the SPH formulation is that the derivative of the function can be simply shifted to the kernel [17]. Hence we can easily compute the gradient of $\mathbf{u}$ as follows:

$$\nabla \mathbf{u}_i = \sum_j \tilde{v}_j \mathbf{u}_{ji} \nabla W(\|\mathbf{x}_i^0 - \mathbf{x}_j^0\|, h) \tag{2.13}$$

The smoothing kernel functions as a way to gradually reduce the effect of neighbouring particles as we get further away from i and thus, it has a great influence on the behaviour of the SPH simulation. The kernel used in [4] [32], originally designed by Solenthaler et al. [5] specifically for improved results in elasticity simulation, is:

$$W(r_{ij}, h) = \begin{cases} c\frac{2h}{\pi} \cos\left(\frac{(r+h)\pi}{2h}\right) + c\frac{2h}{\pi} & 0 \leq r_{ij} \leq h \\ 0 & \text{otherwise,} \end{cases} \tag{2.14}$$

where

$$r_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$$

$$h = \text{influence radius}$$

$$c = \frac{\pi}{8h^4\left(\frac{\pi}{3} - \frac{8}{\pi} + \frac{16}{\pi^2}\right)},$$

assuming an even ($W(r,h) = W(-r,h)$) normalised ($\int W = 1$) smoothing kernel.

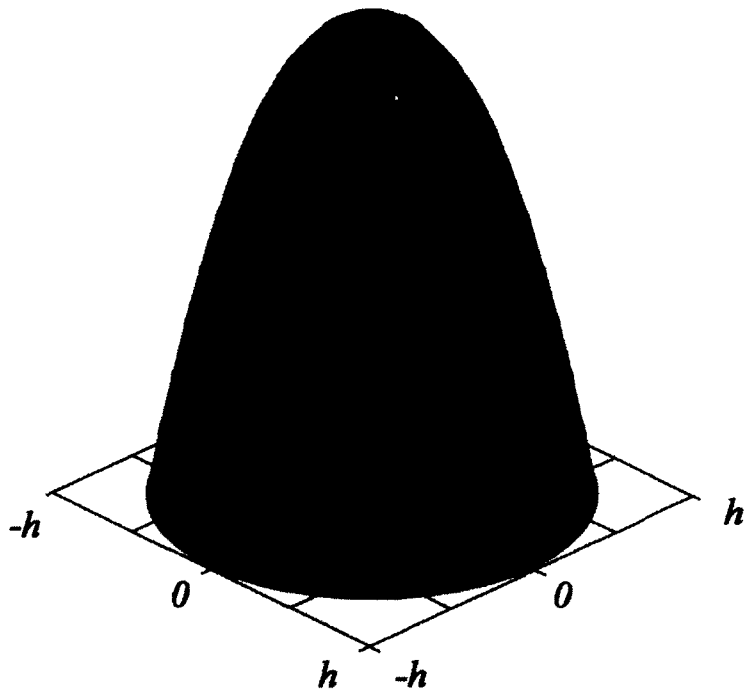

Figure 2.2: The 2D profile of the smoothing kernel function. The light graph demonstrates the spiky shape of $W(r, h)$ and the overlaid darker graph shows its gradient in the direction towards the centre.

Another key characteristic of kernel (2.14) and similar kernels used in other meshless techniques is their so-called "spiky" profile, whose gradient in the direction towards the center becomes the intended smoothing shape used to approximate the gradient of displacement in (2.13). They also have the desirable property of compact support [26], ensuring an absolute zero influence beyond the radius h. This is in contrast to, for example, Gaussian-based kernels, which don't naturally provide a finite influence. Figure 2.2 illustrates a 2D slice of kernel (2.14) and its gradient.

2.3.3 Isolating Particle Rotation

The elastic SPH formulation discussed so far was used by Solenthaler et al. [5] in their unified solver, mixing it successfully with other SPH formulations for plastic, viscoelastic and fluid dynamics. Because the entire simulation was Lagrangian and SPH-based, they were also able to model convincing phase transitions between those different types of dynamics. But since the approximation in (2.13) is only zero-order consistent, the simulation is not rotationally invariant, leading to erroneous handling of rigid rotations, even with the use of a non-linear strain tensor. This meant that if an elastic body was rotating without any internal deformation, the gradient of the displacement field was incorrectly approximated, producing strain, and in turn elastic forces, undesirable in those situations. In some cases, these forces were strong enough to prevent an elastic object from rotating at all. Becker et al. [4] proposed a corotational formulation to fix this problem. The concept is borrowed from successful application of corotational elasticity in FEM simulations for real-time applications [10] and gaming environments [33]. Non-linear strain tensors are rotationally invariant in FEM, and the primary purpose of the corotational formulation was to allow the use of the faster linear strain tensors while correctly handling rotation. Becker's corotated SPH provides rotational invariance for both linear and non-linear strain simulations.

The main idea here is to carefully unrotate the particles back to their original orientation, calculate the displacement field gradient, and then finally rotate them back for corrected force calculation. That is, no matter how the particles are oriented in a deformed state, the rotation is isolated and prevented from affecting the calculation of the true cause of strain, the displacement field gradient, thus giving us a rotationally invariant SPH computation.

For each individual particle, the rigid (affine) transformation matrix $\mathbf{A}_i$ describing the transformation from the reference position to the current deformed position, as

described by Muller's [29] shape matching method, is computed by considering the deformation in the local neighbourhood of particle i, weighted again by the smoothing kernel to reduce the influence of particles further away. For a given particle i:

$$\mathbf{A}_i = \sum_j m_j W(\|\mathbf{x}_i^0 - \mathbf{x}_j^0\|, h) \left((\mathbf{x}_j - \mathbf{x}_i)(\mathbf{x}_j^0 - \mathbf{x}_i^0)^T\right) \tag{2.15}$$

Polar decomposition is then used on matrix $\mathbf{A}$ to eliminate the symmetric part $\sqrt{\mathbf{A}^T\mathbf{A}}$, and obtain the rotational part $\mathbf{R}$:

$$\mathbf{R}_i = \mathbf{A}_i \left(\sqrt{\mathbf{A}_i^T \mathbf{A}_i}\right)^{-1} \tag{2.16}$$

Given each particle's rotation matrix $\mathbf{R}_i$, a rotationally invariant SPH approximation can be obtained. Equation (2.13) is revised and the gradient of the displacement field becomes:

$$\nabla \mathbf{u}_i = \sum_j \tilde{v}_j \mathbf{R}_i^{-1} \mathbf{u}_{ji} \nabla W(\|\mathbf{x}_i - \mathbf{x}_j\|, h) \tag{2.17}$$

2.3.4 Computing Elastic Forces

With $\nabla \mathbf{u}_i$ values in hand, we compute the strain ε_i and stress σ_i for each particle. In our implementation, we use Cauchy's linear strain tensor from equation (2.2), and assuming a Hookean isotropic material, we use the linear constitutive relationship from (2.3). The linear strain energy stored in volume $\tilde{v}_i$ of the particle is this given by:

$$U_i = \frac{1}{2} \int_{\tilde{v}_i} \varepsilon_i^T . \sigma_i d\tilde{v}_i \tag{2.18}$$

and the resulting internal elastic force is then computed as:

$$\mathbf{f} = -\nabla \mathbf{u}_i U_i \tag{2.19}$$

The force computation from equation (2.19) for a particle j acting on particle i is then:

$$\mathbf{f}_{ji} = -\nabla \mathbf{u}_j U_i = -\tilde{v}_i \sigma_i \tilde{v}_j \nabla W(\mathbf{x}_i^0 - \mathbf{x}_j^0, h) \tag{2.20}$$

To compute the total elastic force $\mathbf{f}_i$ at particle i, we sum up all the forces acting on it from every neighbour j as follows:

$$\mathbf{f}_i = \sum_j \frac{-\mathbf{R}_i \mathbf{f}_{ji} + \mathbf{R}_j \mathbf{f}_{ij}}{2}, \tag{2.21}$$

where we pre-rotate the particles back to their deformed formation using $\mathbf{R}$, and to conserve momentum (Newton's third law) the forces $\mathbf{f}_{ji}$ and $\mathbf{f}_{ij}$ are treated symmetrically and in opposite directions.

All that remains is to pass these forces to a suitable simulation solver from Section 2.2.2 which advances (integrates) the particles over the time step Δt to update the velocities and return the new particle positions for the next frame of simulation.

This concludes the purely elastic and non-topologically changing component of the simulation framework.

Chapter 3

Method

In this chapter, we present our approach to realistic tearing simulation and how we incorporate it into the corotated SPH elasticity method outlined previously in section 2.3. We start by describing how fracture is visually encoded using fracture disks and allowed to exist in semi-arbitrary locations, followed by parametrizations that allow control over the visual detail of the fracture pattern without disturbing the underlying physical mechanics and finally, incorporating controllable anisotropic influence on the tearing direction. We apply our tearing modifications while striving for minimal impact on the computational cost of the already efficient elastic simulation.

3.1 Tearing

3.1.1 Fractured Kernel Shape Functions

In FEM or Mass-Spring System (MSS) simulations [26], the discretized simulation nodes have well defined connections that can be removed or split apart to introduce fracture and topological change into the model. Meshless methods like SPH, on the other hand, do not have this kind of explicit connectivity between its smoothed particles. What we can do, however, is modify the weighted kernel shape functions,

such that nodes separated due to fracturing no longer continue to influence the displacement field gradient calculation, even if those nodes are still within each other's neighbourhood radius h.

For this purpose, Baleytschko et al. [34] introduced discontinuities in the shape functions using a visibility criteria. Two nodes within each other's neighboorhoods can no longer interact with each other if the ray passing through them intersects a crack surface. This prevented the shape functions from crossing the crack, but introduced abrupt discontinuities in $\nabla \mathbf{u}$ on both sides of the crack, not just across it. This prompted Organ et al. [35] to introduce transparency to the visibility check, providing a smoother outcome for the modified shape functions and better transitions near the tip of the crack front. We follow this approach and develop our own transparency criteria, motivated by empirical results and the afforded controllability of the tearing style. Ho

Every particle i stores a visibility value $A_{ij} \in [0, 1]$ for every particle j in its neighbourhood, where 0 means the particles can completely see each other and 1 indicates no visibility at all. This is symmetric, so $A_{ij} = A_{ji}$. The value is used to artificially increase the distance between $\mathbf{x}_i^0$ and $\mathbf{x}_j^0$ for the kernel computation in equation (2.14) for both particles. So, the new distance metric used for r becomes:

$$r_{ij} = \|\mathbf{x}_i^0 - \mathbf{x}_j^0\| + A_{ij}h \tag{3.1}$$

A crack surface is defined as a round disk centred at position $\mathbf{c}$ with axis vector $\mathbf{a}$ and radius k. Wherever a fracture event is detected, a new disk is inserted. The criteria for determining the disk location and orientation are discussed in sections 3.1.2-3.2.2. Once inserted, we trigger a visibility check for all nearby particles: for a given particle i, we trace a ray to each of its neighbours j and denote the location of intersection of each ray with the fracture disk as $\mathbf{d}_{ij}$. The opacity α_{ij} at that point

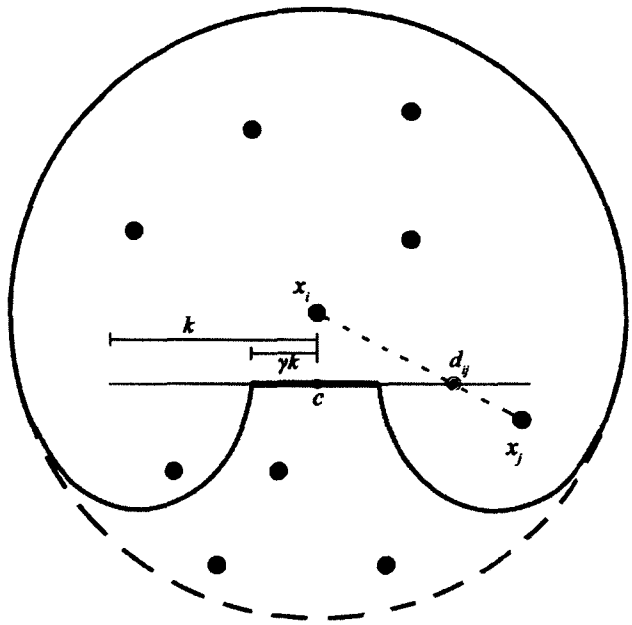

Figure 3.1: Effect of a visibility check through a fractuare disk (red) on the kernel shape function (thick line) with influence radius h (dotted circle).

on the disk gets added to the principle opacity A_{ij}. The opacity $\alpha \in [0,1]$ is radially defined on the disk as:

$$\alpha = \begin{cases} 1 & \|\mathbf{d}_{ij} - \mathbf{c}\| \leq \gamma k \\ 1 - \left(\dfrac{\|\mathbf{d}_{ij} - \mathbf{c}\| - \gamma k}{k(1-\gamma)} \right)^e & \text{otherwise,} \end{cases} \tag{3.2}$$

where $\gamma \in [0,1]$ defines the rigidity of the fracture giving the user control over how the fracture propagates locally. Visually, γk can be thought of as the radius of an inner disk that is completely opaque ($\alpha = 1$) and the rest of the disk gets gradually more transparent as we move closer to the outer edge, radius k, of the disk. Figure 3.1 shows the gradual effect of disk transparency on the kernel shape function for particle i.

The α_{ij} value is the same for both particles ($\alpha_{ij} = \alpha_{ji}$). If multiple disks where inserted in the same time step (frame) of animation, the α_{ij} obtained from all the disk visibility checks are treated additively and A_{ij} is updated before discarding all the disks at the end of the frame. A_{ij} can only increase, as we treat fracture damage as permanent, and once it reaches 1, the particles are dropped from each other's neighbourhood lists and are never used again in each other's $\nabla \mathbf{u}$ calculation. Also, as a general rule, if the particles become too distant from each other $\|\mathbf{x}_i - \mathbf{x}_j\| > 2.5h$, we drop them from their corresponding lists, regardless of A_{ij}. Note that while the visibility check is done using the current deformed state with positions $\mathbf{x}_i$ and $\mathbf{x}_j$, the neighbourhood update is done on the initial state $\mathbf{x}_i^0$ and $\mathbf{x}_j^0$ as that is where the elastic structure is originally defined. High γ values produce very rigid and clean cuts, whereas lower values produce softness in the fracture lines and interesting patterns will arise (Figures 3.2 and 4.1). Of course, γ values can also be defined globally or specified locally per node. We empirically found $\gamma = \frac{1}{2.4}$ to be a suitable default value.

The method presented in this section is a simple user-friendly way to alter the dynamic style of fracture, affording flexibility of design and working alongside (without being restricted by) the physical parameters used for elasticity simulation.

3.1.2 Fracture Criteria

Fracture or tearing is initiated at any point in the body where the maximal eigenvalue of the stress tensor σ exceeds the material threshold [36]. For an isotropic material, the crack is propagated orthogonally to the corresponding eigenvector.

The stress criteria is evaluated at every particle in the simulation and we can directly begin the crack at the location of that node. However, arbitrary crack locations that are not dependant on the particle distribution are also very achievable.

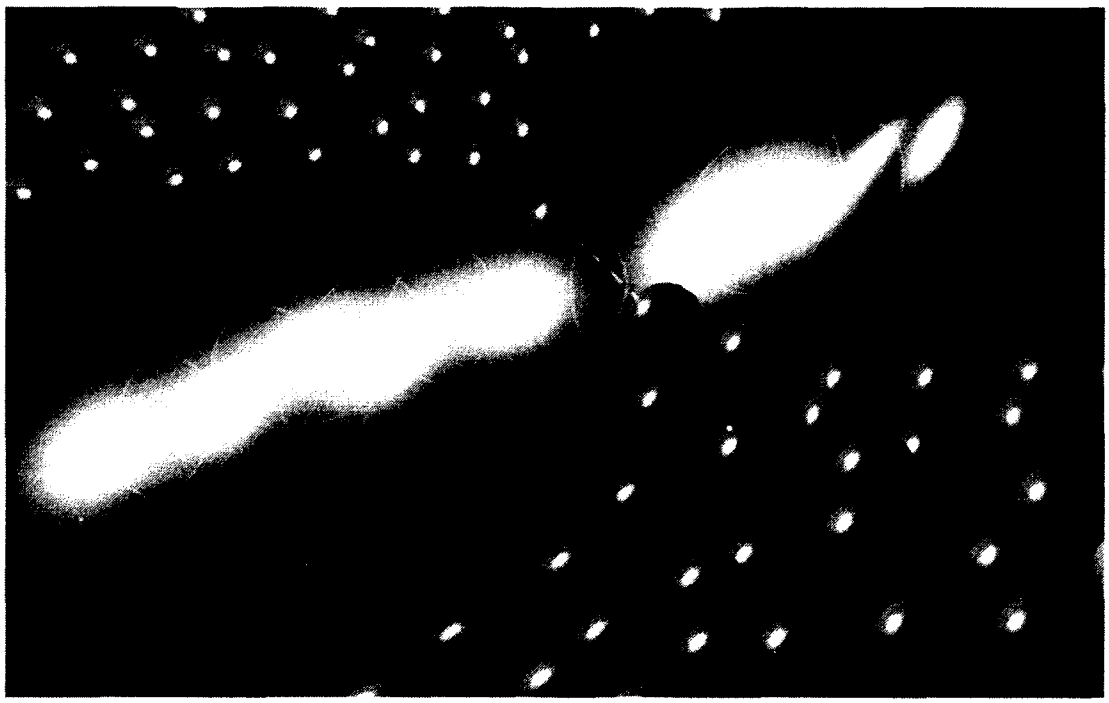

Figure 3.2: Effect of transparent fracture disks on the fracture style. High γ values produce stiff fractures that seem to pop. On the other hand, low γ produces results more akin to soft tearing, with interesting small-scale effects like shown here, where part of the fractured object still seems to be "hanging by a thread."

Part of what makes meshless methods attractive for fracture simulation is their inherent flexibility with resampling, allowing the freedom to insert particles without worrying about proper remeshing like in FEM methods. This is because approximation schemes, like the SPH we presented earlier, rely on the local neighbourhood information of a particle. Inserting new particles becomes a matter of properly redistributing mass (a single scalar value) from the existing neighbouring particles, and the SPH formulation will continue automatically without any adverse effects on the mechanical properties simulated so far. The same concept applies to removing particles from the domain. This adaptivity is heavily used in [37] and [38] to increase simulation samples near surfaces, and to stochastically resample around a detected fractured node to find a more arbitrary location that might have a higher stress value and start their crack propagation from there instead. Lui et al's [21] also use resampling around areas of high stress, before the fracture criteria is exceeded, to increase accuracy of finding a crack location if that area ever fractured.

While our SPH framework allows such flexible resampling schemes, we instead opt for a simpler method that preserves the physics simulation nodes as they are and focuses on approximating a better crack location. Our target is still primarily an interactive application, and although resampling might increase accuracy, it could result in an unpredictable increase in the number of simulation nodes and effectively penalize simulations with a high count of fracturing events.

Our crack location method begins with a node whose principal stress (eigenvalue) ν_i has exceeded the material fracture threshold τ. We obtain the corresponding stress eigenvector σ_{ν_i} and the orthogonal fracture direction $\mathbf{z}_i$. A fracture disk is inserted with $\mathbf{c} = \mathbf{x}_i$, $\mathbf{a} = \sigma_{\nu_i}$ and $k = h + 2(\nu_i - \tau)$. We then split the neighbourhood of particle i into *Left* and *Right* neighbourhoods by the plane containing $\mathbf{z}_i$ and whose normal is in σ_{ν_i}. The weighted average of the maximal stresses is calculated for each

side independently as $\bar{\nu}_{Left}$ and $\bar{\nu}_{Right}$:

$$\bar{\nu}_{Left} = \sum_j m_j \nu_j W(\|\mathbf{x}_j - \mathbf{x}_i\|, h) \text{ , for every } j \text{ to the 'left' of } \mathbf{z}_i \tag{3.3}$$

$$\bar{\nu}_{Right} = \sum_j m_j \nu_j W(\|\mathbf{x}_j - \mathbf{x}_i\|, h) \text{ , for every } j \text{ to the 'right' of } \mathbf{z}_i \tag{3.4}$$

The main idea here is to find out which side is more stressed, and this bias informs us of where the fracture would have most likely occurred. Finally, we perturb the crack location from it's original position $\mathbf{x}_i$ along the stress vector σ_{ν_i} by an amount directly relative to the difference between $\bar{\nu}_{Left}$ and $\bar{\nu}_{Right}$.

Since meshless methods are inherently denser than FEM methods, we find this minimal approach reasonable for finding a physically-motivated arbitrary crack location without sacrificing computational efficiency.

3.2 Anisotropic Tearing

In this section we describe the specifics related to anisotropic control over the tearing direction, using terminology analogous to fibres found in biological tissue [25].

3.2.1 Fibre Orientation Maintenance

In our physics model, we allow a user-controlled vector field over the simulation particles to denote the desired directional influence on the tear propagation. Each vector is called a *fibre*, and is conceptualised as a 1-dimensional thread flowing through the body of the object. We assume that the fibres fill up the 3D domain of the simulation. That is, any particle lying within the continuum of the object has a single fibre passing through it, indicating the preferred direction of tearing at that particle. These threads give strength to the material along their main vector and

resist tearing across it. Before we further discuss how this fibre influences the tearing process, we examine how the fibre is defined in our simulation.

The fibrous structure of the object is discretized into a vector field over our existing set of simulation particles P. Every particle is initialised with the 3D vector θ_i indicating fibre direction through that particle position. The directions can be defined globally, generated procedurally (for example following the curvature of the object), or setup by hand using 3D brushes or other manual techniques. Linear, quadratic or quaternion interpolation can be used to find the intended fibre direction at arbitrary points throughout the domain.

As the simulation progresses and the object is deformed by elastic and external forces, we need to update the direction of the fibre to reflect the true fibre structure in the deformed state, because elastic forces do not produce local torque. This was not a problem in Lui et al's [21] meshless brittle fracture simulation because they used rigid body dynamics and could correctly maintain the fibre direction throughout the simulation. Elastic FEM methods, like [25], can simply embed the fibre within each triangle (2D) or tetrahedron (3D) and track the orientation as the simplex deforms.

We initially experimented with an implicit definition of θ_i, so for a certain particle i, we chose a neighbouring particle j such that $\mathbf{x}_j^0 - \mathbf{x}_i^0$ provided the closest approximation to the user input direction θ_i. Once initialised, this link between i and j is constant throughout the simulation. This method was simple to implement, and gave a very stable non-jittery definition of θ_i but we soon realised its disadvantages. The first obvious problem was that the fibre structure became entirely dependent on the initial distribution of particles, hence we could not truly define arbitrary fibre directions. Additionally, having the direction coupled with a single particle j meant that θ_i was no longer aware of deformations on the opposite side of i and would not update the fibre at i accordingly. Lastly, further maintenance was required when a fracturing event separated i from j, requiring us to find a new suitable neighbour j

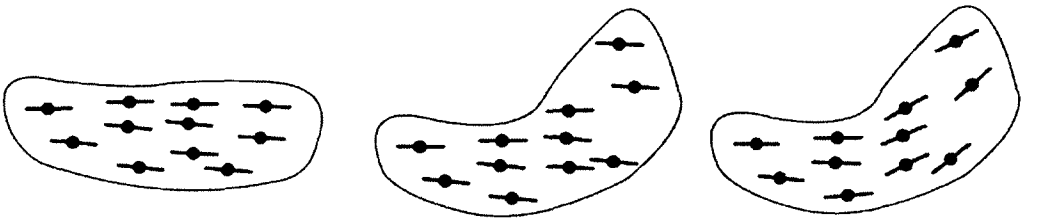

Figure 3.3: Fibre orientation maintenance. From left to right: undeformed particles with initial fibre direction θ_i; deformed particles with no update to fibre direction; deformed particles with explicit fibre update $\phi_i = \mathbf{R}_i\theta_i$

for i. It quickly became apparent that this approach was not robust enough for our tearable simulation.

We opted instead for an explicit definition of θ_i (Figure 3.3). The utility of the SPH corotational formulation was furthered by using the readily available rotation matrix $\mathbf{R}_i$ previously calculated in equation (2.16). This gave us a very stable and efficient computation of the current fibre direction ϕ_i:

$$\phi_i = \mathbf{R}_i\theta_i, \tag{3.5}$$

where θ_i (or $-\theta_i$, we do not differentiate between them) is now just the initial vector definition of the fibre, centred at $\mathbf{x}_i$ and is no longer restricted to the distribution of the particles, and the current ϕ_i is not affected by fracturing events.

3.2.2 Anisotropic Influence of Fibre on the Tearing Direction

In this section we describe how fibre orientation is used to influence the fracture direction $\mathbf{z}$ in 3.1.2, resulting in the desired anisotropic fracture.

Fibres in our simulation are there to resist tearing across the fibre's main direction. Given the plane Φ_i whose normal is ϕ_i, we want to suppress the component of the fracture vector $\mathbf{z}_i$ in the direction of its projection onto Φ_i. The new fracture vector

is then:

$$\mathbf{z}_i' = \mathbf{z}_i - (1 - \beta_i)\text{proj}_{\Phi}(\mathbf{z}_i), \tag{3.6}$$

where β_i is a penalty term deciding the magnitude of the fibre's influence (0 = completely follow fibre direction; 1 = no fibre influence at all). What distinguishes our implementation from previous penalty-based anisotropy models [21] [37] [38] is that in addition to allowing user-input values for β_i, we improve simulation realism by also considering the physical rate of deformation (strain rate $\dot{\varepsilon}_i$) for varying β_i. The faster the rate of deformation is, the less the fibre can say about the tear's direction, hence a higher β_i. The effect of our approach can be seen in Figure 3.4.

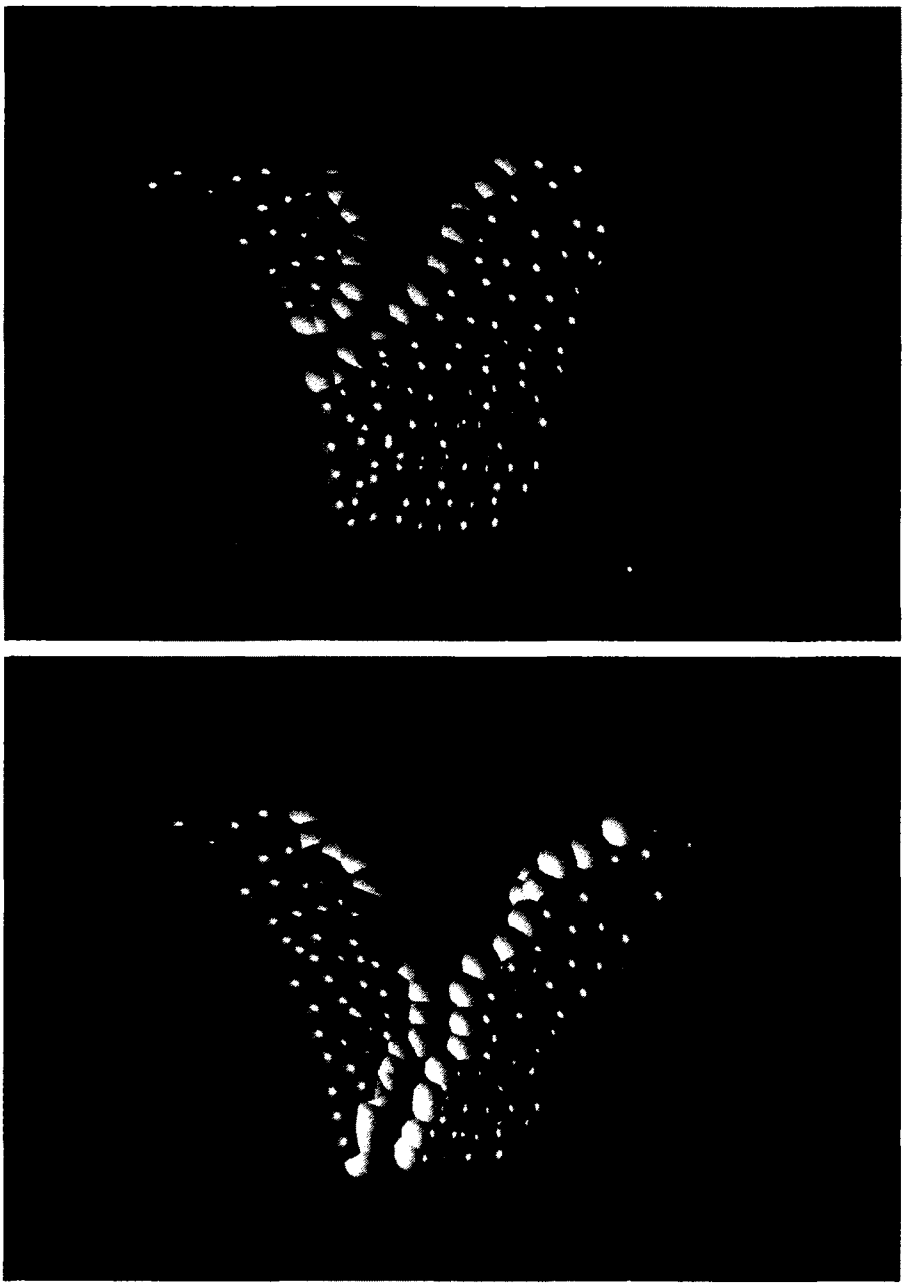

Figure 3.4: An originally rectangular soft body with diagonal fibres pointing north-east being torn apart by external forces (green cones). Top: pealing the object slowly allows the fibre to dictate the tearing direction. Bottom: a faster strain-rate correctly reduces the anisotropic influence of fibre, closely resembling tearing in a fibre-free object

Chapter 4

Results

4.1 Implementation

We implemented our elasticity and tearing models by extending the open source SPH simulator FLUIDSv.2 [39], resulting in a system that can simulate both fluids and tearable elastic bodies. The code is written in C++ and utilizes OpenGL for graphical output. The tests were performed on a Windows client with a dual-core Intel Core2Duo 2.2GHz CPU, an Nvidia GTS250 graphics card, and 6GB of RAM.

The particles in all of our tests are on average 0.05 units apart with an influence radius $h = 0.2$. The number of neighbours a particle can have within its influence radius is limited to the nearest 20. This allows us to use an adjacency list to maintain neighbourhood information quite efficiently. In addition, we inserted our deforming particles into a spatial hash as described in [40] and [41], which is a standard way to accelerate broad-phase collision detection and fluid neighbourhoods, but we also use it to quickly find the grid(s) around an inserted fracture disk, performing the disk visibility check on only those particles.

4.1.1 Sample Pseudocode

Initialization:

1. Fill domain with n particles (Section 3.1)
2. Build neighbourhood lists using $\mathbf{x}^0$ positions
3. Visibility check concave regions of domain
4. Given mass m_i, compute density ρ_i and represented volume $\tilde{v}_i$

Simulation Cycle:

5. for each particle $i \in P$:
6. calculate external forces $f_{i,ext}$ (gravity, collisions, etc.)
7. update neighbourhood list if necessary($A_{ij} > 1$ or $\|\mathbf{x}_i - \mathbf{x}_j\| > 2.5h$)
8. query neighbourhood list and calculate $\nabla \mathbf{u}_i$
9. update fibre direction ϕ_i, calculate strain, stress and elastic force $\mathbf{f}_i$
10. calculate maximal stress eigenvalue ν_i
11. if $\nu_i \geq \tau$:
12. mark particle i as fractured
13. insert fracture disk $\mathbf{d}_i$
14. adjust disk position using stress bias
15. adjust disk orientation using anisotropic fibres
16. for each fracture disk d_i:
17. for each particle i within disk influence ($2.5h$):
18. check visibility and update A_{ij} with each j
19. discard all disks
20. for each particle i:
21. Solve motion equation: send $\mathbf{f}_i$ and $\mathbf{f}_{i,ext}$ to integrate using timestep Δt
22. update position $\mathbf{x}_i$ for next iteration

4.2 Performance and Results

Table 4.1 summarizes the performance for the samples shown in this report. We successfully achieve interactive rates in our tests, with our profiler indicating that our calculations take an average of 62% of the total processing time per frame, with the rest for rendering and other processes. These results, while not distinguished from similar purely elastic simulation models, show considerable speed-up when compared to similar fracturing simulations, in some cases even an order of a magnitude faster, as shown in Table 4.2. We do note that surfacing would incur additional costs, but our physical fracturing simulation speeds already seem promising. We compare our method with other meshless techniques in computer graphics that allow soft-body tearing. We also include the performance of Becker et al. [4], the purely elastic framework that we extended, to show that our tearing method does not present a large computational penalty, even in situations with relatively high particle count. Interestingly, as the number of fractured partitions increased, the simulation got faster due to a reduced neighbourhood list. Figure 4.1 shows an example of an elastic soft body hanging from a static support bar (green) and getting torn by its own weight. Figure 3.4 shows how we enhance the realism of anisotropic tearing by considering the velocity of deformation, strain rate.

As with every physics simulator in computer graphics, stability is an important factor to consider in our tests and implementation design. We used Leapfrog integration, a Verlet variant popular in computer graphics dynamics [26], to advance our simulation, which was fast to solve and provided a high degree of stability. Complex cases benefited from increasing particle samples and decreasing the timestep, at an added computational cost. Generally, SPH fluid pressure forces where also useful to maintain stability for larger time steps [14] [32] [4]. We also found it helpful to spread the fracture disk insertion process over several frames (4 to 10) by scaling the disk

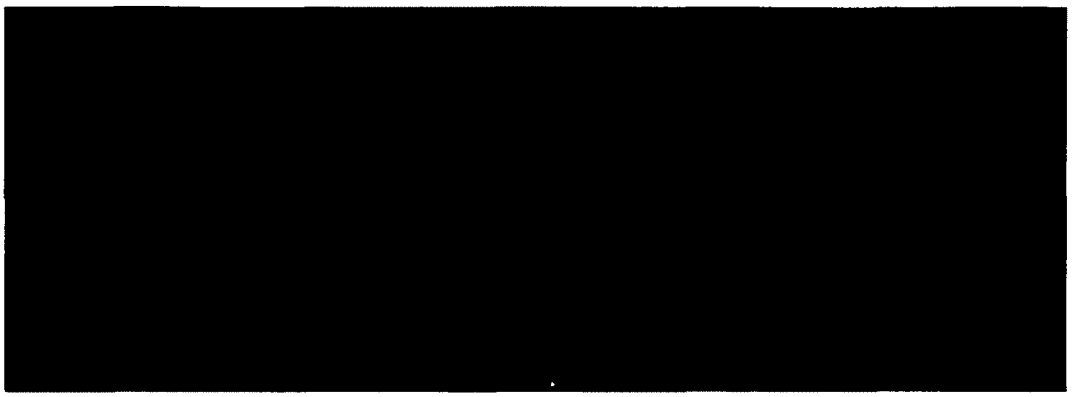

Figure 4.1: Comparing the effects of γ on the tearing pattern, using the same elastic properties. From left to right: a) Initial state ($t = 0sec$); b) $\gamma = 0.85$ ($t = 1sec$); c) $\gamma = 0.3$ ($t = 1sec$)

Sample	# Particles	Tear Rigidity γ	Δt(ms)	Frames/Sec
Figure 3.2	700	0.5	5	65
Figure 3.4	300	0.9	5	86
Figure 4.1b	24000	0.85	5	37
Figure 4.1c	24000	0.3	5	34

Table 4.1: Performance results for samples shown in this report.

size from $0.25k$ to k. Aside from being a reasonable reflection of reality, this had a favourable impact on our computational and visual stability.

As Table 4.1 shows, we can achieve interactive rates ($\geq$ 30fps), but in order to achieve real-time simulation we need to progress at larger time-steps. Possible methods to improve the running time include a common approach in game engines [28] to limit or stop physics calculations if they take too long. However, this primarily applies to collision detection and other constraints that could eventually lead to inaccurate collisions which is, to a degree, acceptable in a gaming environment. But

these types of artefacts would not be tolerated in a surgical training simulator, for example. Instead, it would be more fruitful to pursue a GPU parallel implementation coupled with an implicit integrator. This leverages the independence and parallelism of neighbourhood management and provides stability for much larger time steps and higher particle counts, bringing us closer to our real-time goals.

While evaluating our explicit fibre tracking method (Section 3.2.1), we observed that temporal coherency of the fibre orientation was not perfect (i.e. it seamed a bit jittery) for particles near the surface of newly created fractures. We believe this happens because the SPH-based rotation estimation $\mathbf{R}$ has lesser accuracy due to a reduced number of neighbouring particles compared to the neighbours at the initial reference positions. The effect on the simulation, however, was negligible and the method is overall very sufficient for a fairly accurate and robust maintenance of fibre direction. We also noticed that it performs quite well with primarily dilation-based deformations (stretch and compression), however it struggles to maintain the fibre orientation accurately when the object is subjected to large shear deformations (Figure 4.2). We attempted, unsuccessfully, to remedy this by extracting the local shear information from the Jacobian of the displacement field to inform the fibre orientation. We suspect that a mixed explicit-implicit fibre maintenance method would help greatly in this situation.

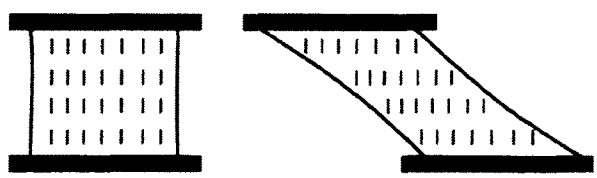

Figure 4.2: Very large rotation-less shear deformations produce inaccurate fibre orientations, possibly resulting in jagged and disconnected tearing patterns.

Method	Sample Description	# Particles	Time/frame (ms)
Muller 2004 [14]	MLS-based; Max Plank model example.	400	35
Pauly 2005 [20]	MLS-based, specifically designed for fracture simulation; Gum-under-shoe example.	2200	360
Liu 2011 [21]	MLPG-based rigid fracture; Coffee plate example.	4857	52010
Gerszewski 2009 [15]	Particle-based elasto-plasticity w/out initial rest state; Bunny splitting on sphere example.	40556	53069
Becker 2009 [4]	SPH-based elasticity, no tearing; Tested scenario from Figure 4.1a.	24000	25
Our Method	SPH-based elasticity with tearing; Figure 4.1b.	24000	29

Table 4.2: Comparison of the physics processing time per frame between various meshless methods that allow for soft-body tearing.

Chapter 5

Conclusion

In this thesis, we have presented a method for soft-body tearing that emphasizes controllability, speed and stability without compromising realism and accuracy, all of which are desired aspects in game, film and medical simulation development. We provided simple parametrizations for two interesting phenomenons in tearing simulation. The first, controls how clean or rigid the cuts are; the second involved anisotropic control of the tearing direction for mechanically isotropic materials using embedded fibres in the deformable body.

Our approach to tearing was motivated by the need for control of the fracturing phenomenon without necessarily modifying the underlying mechanics of the elastic behaviour and we find the physically-informed variety afforded by our method to be a suitable solution to satisfying those needs. We showed how to incorporate our approach into an existing soft-body simulator by extending an elastic solids SPH framework, acknowledging the versatility of meshless methods in fracturing simulations and their suitability for extension into unified physics solvers. Our implementation demonstrated the real-time potential of our approach, and validated the choice of framework and the simplicity of the method.

5.1 Future work

We conclude with a brief discussion of a few notable future directions:

- Focusing more on realism for medical soft tissue which exhibits highly non-linear and anisotropic behaviour [23, 24] which is difficult to simulate and tear accurately in real-time. This would involve extending our anisotropic component to model more sophisticated fibre structures including grids and strips, in addition to accounting for anisotropy, not just in tearing, but in the mechanical behaviour itself where the material stress thresholds vary across different directions. Existing real-time 2D FEM anisotropic tearing [25] models look promising and we are interested in applying similar concepts in a 3D meshless framework while avoiding FEM's non-trivial problems with remeshing.

- We are very encouraged after seeing our elastic results to try and incorporate plasticity [42] into our simulation and reckon that we would be able to achieve an even higher degree of control over the tearing style using our transparent fracture disk formulation (Figure 5.1). Elastic forces generate energy in the deformed body while plastic flow dissipates it, directly affecting the fracture potential and its ductility.

- Finally, we would like to work on an efficient GPU implementation to accelerate our physics simulation and more importantly to clear up more processing time for other elements of the interactive environment like haptic feedback, surfacing and high quality rendering [43].

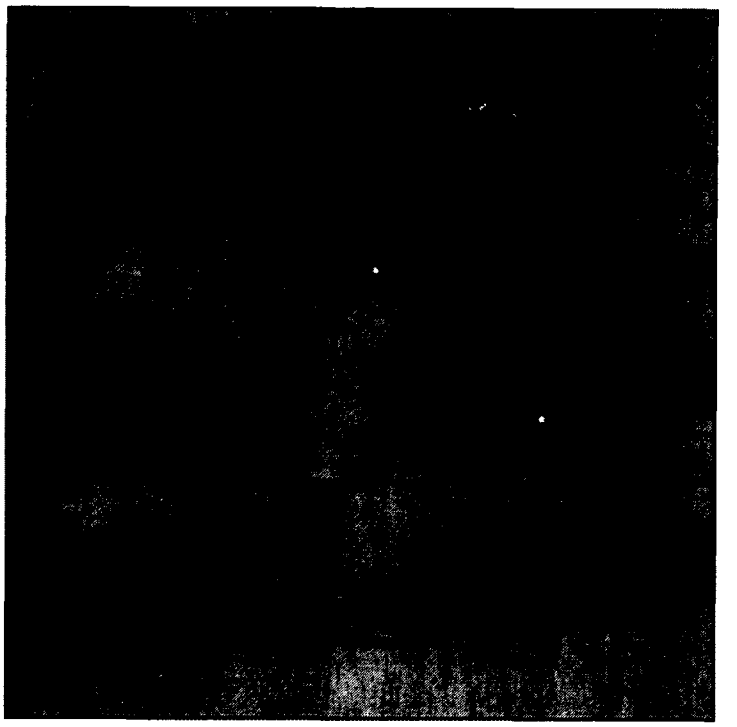

Figure 5.1: The type of fracture control we believe is achievable when we introduce plastic flow into our current framework.

References

[1] U. Meier, O. López, C. Monserrat, M. C. Juan, and M. Alcañiz. "Real-time deformable models for surgery simulation: a survey." *Computer methods and programs in biomedicine* **77**(3), 183–97. ISSN 0169-2607 (2005).

[2] N. Famaey and J. Vander Sloten. "Soft tissue modelling for applications in virtual surgery and surgical robotics." *Computer methods in biomechanics and biomedical engineering* **11**(4), 351–66. ISSN 1025-5842 (2008).

[3] M. Mahvash and a.M. Okamura. "A Fracture Mechanics Approach to Haptic Synthesis of Tissue Cutting with Scissors." *First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems* pages 356–362 (2005).

[4] M. Becker and M. Teschner. "Corotated SPH for deformable solids." *Computer* (2009).

[5] B. Solenthaler, J. Schläfli, and R. Pajarola. "A unified particle model for fluidsolid interactions." *Computer Animation and Virtual Worlds* **18**(1), 69–82. ISSN 15464261 (2007).

[6] D. Terzopoulos. "Modeling Inelastic Deformation : Viscoelasticity , Plasticity , Fracture." *Computer* **22**(4), 269–278 (1988).

[7] J. F. O'Brien and J. K. Hodgins. "Graphical modeling and animation of brittle fracture." In "Proceedings of ACM SIGGRAPH 1999," pages 137–146. ACM Press/Addison-Wesley Publishing Co. (1999).

[8] J. F. O'Brien, A. W. Bargteil, and J. K. Hodgins. "Graphical modeling and animation of ductile fracture." In "Proceedings of ACM SIGGRAPH 2002," pages 291–294. ACM Press (2002).

[9] G. Irving, J. Teran, and R. Fedkiw. "Invertible finite elements for robust simulation of large deformation." In "Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation," SCA '04, pages 131–140. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland. ISBN 3-905673-14-2 (2004).

[10] M. Müller and M. Gross. "Interactive virtual materials." In "Proceedings of Graphics Interface 2004," GI '04, pages 239–246. Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada. ISBN 1-56881-227-2 (2004).

[11] E. G. Parker and J. F. O'Brien. "Real-time deformation and fracture in a game environment." In "Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation," pages 156–166 (2009).

[12] C. Wojtan and G. Turk. "Fast viscoelastic behavior with thin features." *ACM Trans. Graph.* **27**, 47:1–47:8. ISSN 0730-0301 (2008).

[13] M. Wicke, D. Ritchie, B. M. Klingner, S. Burke, J. R. Shewchuk, and J. F. O'Brien. "Dynamic local remeshing for elastoplastic simulation." In "Proceedings of ACM SIGGRAPH 2010," pages 49:1–11 (2010).

[14] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. "Point based animation of elastic, plastic and melting objects." In "Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation," SCA '04, pages 141–151. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland. ISBN 3-905673-14-2 (2004).

[15] D. Gerszewski, H. Bhattacharya, and A. W. Bargteil. "A point-based method for animating elastoplastic solids." *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation - SCA '09* **1**, 133 (2009).

[16] R. A. Gingold and J. J. Monaghan. "Smoothed particle hydrodynamics: theory and application to non-spherical stars." *Monthly Notices of the Royal Astronomical Society* **181**(2), 375–389. ISSN 00358711 (1977).

[17] M. Müller, D. Charypar, and M. Gross. "Particle-based fluid simulation for interactive applications." In "Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation," SCA '03, pages 154–159. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland. ISBN 1-58113-659-5 (2003).

[18] J. Tan and X. Yang. "Physically-based fluid animation: A survey." *Science in China Series F: Information Sciences* **52**, 723–740. ISSN 1009-2757. 10.1007/s11432-009-0091-z (2009).

[19] B. Adams, M. Pauly, R. Keiser, and L. J. Guibas. "Adaptively sampled particle fluids." *ACM Trans. Graph.* **26**. ISSN 0730-0301 (2007).

[20] M. Pauly, R. Keiser, B. Adams, P. Dutré, M. Gross, and L. J. Guibas. "Meshless Animation of Fracturing Solids." *ACM transactions on graphics* **24**(3), 957–964. ISSN 0730-0301 (2005).

[21] N. Liu, X. He, S. Li, and G. Wang. "Meshless simulation of brittle fracture." *Computer Animation And Virtual Worlds* (April), 115–124 (2011).

[22] H. Meyerhenke, B. Monien, and S. Schamberger. "Graph partitioning and disturbed diffusion." *Parallel Comput.* **35**, 544–569. ISSN 0167-8191 (2009).

[23] S. Niroomandi, I. Alfaro, E. Cueto, and F. Chinesta. "Real-time deformable models of non-linear tissues by model reduction techniques." *Computer methods and programs in biomedicine* **91**(3), 223–31. ISSN 0169-2607 (2008).

[24] T. Zohdi. "A computational framework for network modeling of fibrous biological tissue deformation and rupture." *Computer Methods in Applied Mechanics and Engineering* **196**(31-32), 2972–2980. ISSN 00457825 (2007).

[25] J. Allard, M. Marchal, and S. Cotin. "Fiber-based fracture model for simulating soft tissue tearing." *Studies in health technology and informatics* **142**, 13–8. ISSN 0926-9630 (2009).

[26] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson. "Physically Based Deformable Models in Computer Graphics." *Computer Graphics Forum* **25**(4), 809–836. ISSN 0167-7055 (2006).

[27] K. Erleben, J. Sporring, K. Henriksen, and K. Dohlman. *Physics-based Animation (Graphics Series)*. Charles River Media, Inc., Rockland, MA, USA. ISBN 1584503807 (2005).

[28] G. van den Bergen and D. Gregorius. *Game Physics Pearls*, chapter 2. A K Peters Ltd. (2010).

[29] M. Müller, B. Heidelberger, M. Teschner, and M. Gross. "Meshless deformations based on shape matching." *ACM Trans. Graph.* **24**, 471–478. ISSN 0730-0301 (2005).

[30] M. Müller and N. Chentanez. "Solid simulation with oriented particles." *ACM SIGGRAPH 2011 papers on - SIGGRAPH '11* **1**(212), 1 (2011).

[31] Q. Du, V. Faber, and M. Gunzburger. "Centroidal voronoi tessellations: Applications and algorithms." *SIAM Rev.* **41**, 637–676. ISSN 0036-1445 (1999).

[32] D. Gerszewski, H. Bhattacharya, and A. W. Bargteil. "A point-based method for animating elastoplastic solids." *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation - SCA '09* **1**, 133 (2009).

[33] M. Müller, J. Stam, D. James, and N. Thürey. "Real time physics: class notes." In "ACM SIGGRAPH 2008 classes," SIGGRAPH '08, pages 88:1–88:90. ACM, New York, NY, USA (2008).

[34] T. Belytschko, H. Chen, J. Xu, and G. Zi. "Dynamic crack propagation based on loss of hyperbolicity and a new discontinuous enrichment." *International Journal for Numerical Methods in Engineering* **58**(12), 1873–1905 (2003).

[35] D. Organ, M. Fleming, T. Terry, and T. Belytschko. "Continuous meshless approximations for nonconvex bodies by diffraction and transparency." *Computational Mechanics* **18**, 225–235 (1996).

[36] T. L. Anderson. *Fracture Mechanics: Fundamentals and Applications*, volume 2. CRC Press. ISBN 9780849316562 (1995).

[37] M. Pauly, R. Keiser, B. Adams, P. Dutré, M. Gross, and L. J. Guibas. "Meshless Animation of Fracturing Solids." *ACM transactions on graphics* **24**(3), 957–964. ISSN 0730-0301 (2005).

[38] N. Molino, Z. Bao, and R. Fedkiw. "A virtual node algorithm for changing mesh topology during simulation." In "ACM SIGGRAPH 2005 Courses," SIGGRAPH '05. ACM, New York, NY, USA (2005).

[39] R. C. Hoetzlein and T. Höllerer. "Analyzing Performance and Efficiency of Smoothed Particle Hydrodynamics Categories and Subject Descriptors." .

[40] M. Teschner, B. Heidelberger, M. Mueller, D. Pomeranets, and M. Gross. "Optimized spatial hashing for collision detection of deformable objects." pages 47–54 (2003).

[41] G. Viccione, V. Bovolin, and E. P. Carratelli. "Defining and optimizing algorithms for neighbouring particle identification in sph fluid simulations." *International Journal for Numerical Methods in Fluids* **58**(6), 625–638. ISSN 1097-0363 (2008).

[42] J. F. O'Brien, A. W. Bargteil, and J. K. Hodgins. "Graphical modeling and animation of ductile fracture." *ACM Transactions on Graphics* **21**(3), 291–294. ISSN 07300301 (2002).

[43] M. Leflar, O. Hesham, and C. Joslin. "Use of high dynamic range images for improved medical simulations." In N. Magnenat-Thalmann, editor, "Modelling the Physiological Human," volume 5903 of *Lecture Notes in Computer Science*, pages 199–208. Springer Berlin / Heidelberg. ISBN 978-3-642-10468-8. 10.1007/978-3-642-10470-$1_1$7(2009).