Efficient Computational Methods for Physically-based Simulation

Dissertation

der Fakultät für Informations- und Kognitionswissenschaften

der Eberhard-Karls-Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von

Dipl.-Inform. Bernhard Thomaszewski

aus Osterholz-Scharmbeck

Tübingen
2010

Tag der mündlichen Qualifikation:	14.07.2010
Dekan:	Prof. Dr.-Ing. Oliver Kohlbacher
1. Berichterstatter:	Prof. Dr.-Ing. Dr.-Ing. E.h. Wolfgang Straßer
2. Berichterstatter:	Prof. Dr. François Faure (Université Joseph Fourier, Grenoble, INRIA Rhône-Alpes)
3. Berichterstatter:	Prof. Dr. Raquel Urtasun (Toyota Institute of Technology, Chicago)

Pour Solenn

Abstract

Physically-based simulation has become an indispensable foundation of modern entertainment. Animated movies and video games are traditional areas of application, but also contemporary feature films rely more and more on physics-based computer animation. The computational methods used for simulating deformable surfaces and solids, fluids, rigid bodies and their mutual interaction have matured significantly over the last two decades. However, the expectations of spectators and users grow as well, creating a sustained demand for increased realism, better control, higher efficiency and new visual effects. This thesis contributes to these objectives in several ways and proposes new computational methods for cloth simulation, collision handling and rigid body interaction.

Simulating the behavior of cloth and clothing in an accurate and efficient way requires a solid computational framework, which is provided in Chapter 2. Particular attention is paid to an accurate modeling of in-plane deformations, which is achieved with a geometrically nonlinear approach based on continuum mechanics and finite elements.

Building on this basis, Chapter 3 addresses the problem of how to model the complex material behavior of textiles, which typically exhibit nonlinear and direction-dependent properties. Modeling these properties in full detail is computationally expensive, but overly simple approximations fail to capture important characteristics of cloth. In order to resolve these shortcomings, a novel approach is proposed that combines simple elastic materials and continuum-based deformation constraints into a biphasic, anisotropic model of cloth.

Modeling internal properties is only one concern when simulating cloth, another being how to solve the dynamics over time. Striving for fast computation times, many current methods rely on implicit integration and large step sizes to solve a linearized version of the equations of motion. This combination is, however, known to suffer from numerical dissipation, which suppresses surface details and damps motion in an uncontrollable way. Chapter 4 describes a different approach that leverages asynchronous explicit time integration to produce low-damped cloth simulations.

Realism and complexity of cloth animations are typically conveyed through detailed wrinkles and folding patterns, often with many layers of fabric in close proximity. Handling the arising collisions in a robust way is a challenging task that can easily consume the largest part of the computation time. This problem is addressed in Chapter 5, which proposes a method to accelerate collision handling by exploiting the processing power of parallel architectures.

Deformable surfaces interact with their environment through collisions and contact. This is also the most common form of interaction among rigid bodies, which typically dominate the scene in interactive applications. The last chapter of this work introduces a method that extends the range of possible interactions among rigid bodies to magnetic forces and torques. The presented approach is not only accurate but also computationally efficient enough to allow simulations of magnetic rigid bodies at interactive rates.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisor Prof. Dr.-Ing. Dr.-Ing. E.h. Wolfgang Straßer. I feel fortunate for having enjoyed his trust, belief and support throughout my university and doctoral studies.

I would also like to thank Prof. Dr. François Faure and Prof. Dr. Raquel Urtasun who kindly agreed to be the external examiners for my thesis. I am furthermore grateful to Prof. Dr. Eitan Grinspun for inviting me to his lab, for the many insightful discussions as well as for his valuable advice.

This thesis benefits from the knowledge and experience of many former members of the WSI/GRIS and I would like to thank Drs. Olaf Etzmuß, Michael Hauth, Michael Keckeisen, Stefan Kimmerle and Prof. Dr. Markus Wacker. I am particularly indebted to my co-authors Drs. Wolfgang Blochinger, Andreas Gumann, Johannes Mezger and to Simon Pabst. I am also thankful to my student collaborators Fritz Hirt, Markus Huber, Artur Koch, David Mack and Gerhard Prilmeier, whom I had the pleasure to advise during their student respectively diploma theses. Furthermore, I would like to thank Rony Goldenthal for sharing ideas and problems as well as Dr. Günter Knittel for uncounted technical discussions.

I am particularly grateful to Constanze Christ, who, with her organizational skills, protected me from most administrative burdens and, with her kindness and cheerfulness, contributed significantly to the friendly atmosphere in the lab.

Many friends and colleagues made my time at the WSI/GRIS socially enjoyable and I would especially like to thank Peter Biber, Jörg Edelmann, Jan Fischer, Sven Fleck, Benjamin Huhle, Philipp Jenke, Marta Kersten, Robert Kuchar, Florian Liefers, Roman Parys, Aydin Can Polatkan, Anxo del Rio, Timo Schairer and Ralf Sonntag for sharing food, beverages and the pleasures of modern entertainment.

Finally, I owe infinite thanks to my dear wife Solenn for supporting me with loving patience and indulgence throughout my thesis – and beyond.

Contents

List of Figures

Chapter 1

Introduction

Physically-based simulation is one of the most fascinating disciplines of computer graphics, intersecting with diverse scientific fields and having numerous areas of applications. As the most prominent one, physics-based methods have become a vital part of the entertainment industry and are now ubiquitous in video games, animated movies as well as regular feature films, to name but a few. In particular, stunning visual effects involving animations of fluids, cloth, deformable solids, rigid bodies and their mutual interaction are indispensable components of many contemporary productions. In this context, physically-based simulations offer a level of realism and complexity that is far beyond the capabilities of traditional animation techniques such as keyframing.

Producing realistic yet intriguing animations requires powerful computational methods that leverage tools and techniques from many different fields including computational physics, mechanical engineering, applied mathematics, and numerical analysis. While there are many parallels to technical simulations in engineering, there are also substantial differences. For example, crashworthiness codes are employed to deduce safety parameters for automobile construction and material properties for load-bearing structures are determined on the basis of numerical simulations. Clearly, accuracy and fidelity to real world conditions are of paramount importance in this setting. By contrast, physically-based simulations in computer graphics put special emphasis on visual quality, perceived realism, directability and control as well as computational efficiency. This does not mean that physical reality should not be tracked closely, but rather implies that higher accuracy without visual impact is not enough justification for more expensive computations. Due to these different ambitions, progress in physically-based simulation requires ingenuity and inventive adaptation rather than mere adoption and, consequently, a plethora of dedicated methods has emerged from computer graphics. Thanks to these, computer animations have transitioned from exotic and clearly artificial effects to supportive and complementing techniques that achieve seamless blending with real world footage.

The prevalence of computer animation is continuously on the rise, but so are expectations among the audience, which becomes harder to impress and more sensitive (or skeptical) towards the quality of computer generated content and its integration. Consequently, there is a continued demand for increased realism, better control, higher efficiency and new visual effects. While this is a primary motivation for research into physically-based simulation in general, this thesis is in particular concerned with the fields of cloth simulation, collision handling for deformable surfaces, and rigid body interaction.

1.1 Background and Motivation

Thin deformable surfaces are ubiquitous in our environment. Human skin and membrane tissue are relevant to facial animation and virtual surgery while **cloth simulation** is becoming increasingly important for advertisement and film, computer games and also Internet sales. As a particular property of thin surfaces, the resistance to in-plane deformation is typically much higher than resistance to bending. Consequently, compressive in-plane loadings lead to instantaneous lateral deflections, which manifest as diamond-shaped buckling patterns for metal cans or the typical folds that give textiles their characteristic appearance. Compression is therefore hardly observed in cloth, but also stretch deformations occur only to a limited extent. Woven cloth, in particular, is readily stretched by a few percent but beyond a certain threshold, the material becomes stiff and strongly resists further deformation. This *biphasic* behavior can be explained by the underlying weave structure: the range of small deformations corresponds to the straightening of yarns, whose actual material resistance is only exerted after this process. The heterogeneous fabric structure also gives rise to further material diversity, since the weave pattern can be asymmetric with respect to its main directions, referred to as weft and warp, and the yarns running in these directions can have different material properties as well. This leads to *anisotropic* behavior with considerably different stretch resistance in weft and warp directions and a typically much lower resistance to shear deformation. From a practical point of view, modeling this complex and highly nonlinear behavior to full accuracy entails a number of difficulties and requires computationally expensive methods for robust simulation. Approximate techniques are needed in this context, but despite their widespread use, mass-spring systems with linear material laws are not sufficiently accurate to capture the anisotropic and biphasic properties of cloth. We propose an alternative approach which overcomes these problems by combining continuum-based elastic forces and deformation constraints.

Existing approaches for cloth simulation differ greatly in terms of the mechanical model, but eventually all methods lead to discrete equations of motion that have to be solved numerically by means of **time integration**. Implicit schemes such as backward Euler are preferred for their remarkable stability properties, which allow them to use large step sizes and thus skip fine-level and rapidly varying solution details in favor of computational efficiency. This is a mixed blessing, as it offers both *'a great potential for efficient use, as well as a great danger of misuse'*[1]:

[1] quote from Ascher and Petzold [AP98]

striving for fast computation times, many approaches rely on semi-implicit methods that solve only a linearized version of the problem. But when combined with large time steps, this approach is known to suffer from severe numerical dissipation. This is a type of artificial damping that leads to a lack of dynamics and suppresses the formation of fine level details such as small folds and wrinkles. Using smaller time steps reduces this effect, but it also partly defeats the purpose of using implicit integration methods. On the other hand, explicit methods do not suffer from uncontrollable damping, but are typically overlooked due to their limited stability and thus efficiency. In order to improve on this, we explore an alternative approach that is based on asynchronous explicit time integration.

Visually appealing cloth animations often exhibit complex folding patterns with several layers of fabric in close proximity or contact. Furthermore, practical applications typically also involve interactions with static or moving objects. It is needless to say that, for physical plausibility, intersections between cloth and itself or other objects must be prevented under all circumstances. Detecting collisions and preventing intersections is indeed a challenging problem – to which we collectively refer as **collision handling –** and robust methods are notorious for intensive computations. Despite hierarchical methods and numerous extensions towards higher efficiency, collision handling remains a major bottleneck in cloth simulations. A promising direction for accelerating computations is to leverage the processing power of modern *parallel* computers, which is the approach taken in this work. Good parallel efficiency requires a maximum utilization of the available resources. For this purpose, the problem has to be decomposed into a number of subproblems in order to distribute work evenly among processing units. This is, however, a difficult problem in the context of collision handling since the number and locations of collisions change in a dynamic manner and cannot be predicted in advance. The approach presented in this thesis achieves high parallel efficiency without compromising the performance of the sequential algorithm.

Virtual environments typically feature only a small number of deformable objects such as cloth, solids, or fluids while the largest part of the surroundings consists of rigid bodies. Especially in video games, **rigid body interactions** contribute significantly to perceived realism and the level of user immersion. The dynamics of rigid bodies have been studied intensively in celestial mechanics, robotics and computer graphics. Diverse forms of interactions for rigid bodies have been explored, including collisions and frictional contact, external forcing or coupling with other physical objects. But despite the fact that many rigid bodies consist of ferromagnetic materials, *magnetic interaction* has been largely overlooked in graphics. Magnets exert forces and torque on each other and their complex interplay leads to astounding effects which can greatly enrich rigid body animations. This is particularly interesting for applications such as video games and film, but also pedagogy can benefit from didactic tools that help to understand the complex nature of magnetism and its many facets. User interaction is indispensable for such applications and a computational method for magnetic interaction has to be able to provide simulations at interactive rates while being physically accurate. The method presented in the last part of this thesis constitutes the first approach in this direction.

1.2 Overview and Contributions

The following paragraphs provide an overview of this thesis and summarize its main contributions.

Physical Cloth Simulation (Chapter 2) The second chapter describes a computational framework for physical cloth simulation, which serves as a basis for the developments presented in Chapters 3-5. The anisotropic nature of woven fabric demands for an accurate modeling of in-plane deformations and we resort to continuum mechanics and finite elements for this purpose. Since large element rotations have to be expected, we choose a geometrically nonlinear approach but remain with linear triangle elements in order to limit computational costs. A simple material law with a linear stress-strain response is proposed in order to complement the deformation constraints introduced in Chapter 3. We furthemore describe methods for numerical time integration and collision handling, which will be used and extended in the subsequent chapters.

Deformation Constraints for Biphasic Anisotropic Cloth (Chapter 3) The third chapter presents a novel approach for simulating anisotropic biphasic cloth, to which we refer as Continuum-based Strain Limiting (CSL) [TPS09]. The central idea of this method is to combine a weakly-elastic material with geometric constraints that prevent excessive deformation. This model captures the characteristic material behavior of general textiles better than linear-elastic models while remaining simple and computationally efficient. Previous biphasic methods limited deformation along mesh edges, but this is insufficient to reflect the anisotropic properties of common textiles. By contrast, CSL imposes constraints on a continuum-based deformation measure, which allows the enforcement of individual thresholds for stretching in weft and warp directions as well as shearing, thus providing accurate control over deformation. Discrete deformation constraints are formulated in terms of the co-rotational strain tensor, which is discretized using linear triangle finite elements. We cast local deformation limiting on triangles as a (6×6) linear system and propose an efficient method for its solution, which uses matrix decomposition and precomputations to reduce run time costs.

Asynchronous Cloth Simulation (Chapter 4) Combining semi-implicit time integration with large step sizes allows for fast cloth simulation, but animations produced in this way are often plagued by severe artificial damping. In the fourth chapter of this thesis, we propose an alternative approach [TPS08] that is based on explicit asynchronous time stepping. The explicit nature of the time integrator, which was introduced by Lew et al. [LMOW03], effectively eliminates concerns with numerical dissipation, thus enabling simulations of low-damped cloth. Its asynchronous nature allows each element to have its own step size, thus eliminating the strict global restrictions of synchronous explicit methods. Simulating cloth on this basis requires a specific treatment of collisions and, for this purpose, we propose a three-stage strategy that combines synchronous and asynchronous collision handling into a robust and efficient algorithm. We further leverage the

asynchrony of the solver in order to monitor and limit deformations at run time, thus increasing its stability and efficiency.

Parallel Collision Handling (Chapter 5) Although previous work has considered cloth simulation on PC clusters, a dedicated method for parallel collision handling has not been reported in this context. Collision handling is an irregular problem for which static problem decomposition is insufficient to achieve well-balanced workloads. We therefore propose a task-parallel approach with fully dynamic problem decomposition. The sequential basis for collision handling is formed by state-of-the-art implementations of bounding volume hierarchies and impulse-based collision response. We cast recursive hierarchy tests as a depth-first traversal of the corresponding recursion tree in order to dynamically generate parallelism. This strategy is implemented using multithreaded programming and a distributed task pool model allows dynamic load balancing among processing units. Our method was developed to run on distributed-memory architectures [TB06a, TB07] but is readily extended to shared-memory machines [TPB07, TPB08].

Magnetic Interaction for Rigid Body Simulations (Chapter 6) The last chapter of this work describes a computational method for magnetic interaction in rigid body simulations [TGPS08]. Central to this approach, we derive a model for discrete dipole interaction that allows for fast computations of magnetic fields, forces and torques. We decompose arbitrarily shaped objects into aggregates of *dipole cells* to obtain discrete approximations of magnetic fields. Using dipole expansions for both fields and forces leads to a symmetric approach which automatically conserves linear and angular momenta. Accuracy can be controlled by the number of dipole cells used to approximate the magnetic field of a given object. Since a fixed sampling density is wasteful, we exploit the rapid decay of the dipole field in order to construct an adaptive sampling strategy based on a multi-resolution object partitioning. This greatly improves computational efficiency and allows us to simulate dozens of magnetic objects at interactive rates.

1.3 Context and Related Work

This section gives an overview of the existing and related research on physically-based simulation of cloth, solids and rigid bodies. Its intent is to provide a context for the subjects treated in this thesis while more detailed and technical discussions of relevant works are postponed to the corresponding parts of later chapters.

1.3.1 Cloth Simulation

The physically-based simulation of cloth has been an active research area of computer graphics for more than two decades now. Consequently, there is an abundant body of literature which cannot be covered in full here and we will restrict considerations to the subset with most immediate relevance.

Physical Models The earliest work concerned with cloth modeling in computer graphics is due to Weil [Wei86], who used catenary curves to model the static folding patterns of cloth supported by a number of fixed points. While this method can be considered as physically inspired, it is mentioned here rather for historical reasons. In the following, we will not comment on procedural or heuristically motivated approaches and restrict considerations to physically-based methods[2].

Probably the simplest and most direct way of modeling deformable objects is by means of mass-spring systems, which were first used by Platt and Badler [PB81] to animate facial expressions. The basic idea is to sample the surface (or volume) of interest with a set of point masses (or particles) linked by springs or some other type of interaction potential. Breen et al. [BHW94] used energy functions based on measured data to simulate the drape of woven cloth with a regular grid of particles. This approach was originally limited to the static case, but was later extended to the dynamic setting by Eberhardt et al. [EWS96]. In contrast to these rather complex particle interactions, Provot [Pro95] described a method that uses only simple, linear springs. The latter are arranged on a regular grid in order to resist stretching and compression (along edges), shearing (along cell diagonals), and bending (two consecutive collinear edges).

Mass-spring and particle systems, collectively referred to as direct methods, are popular for easy implementation, but a major drawback is that they require regular discretizations. This is a significant limitation when dealing with complex garments, which are best modeled with unstructured triangle meshes. Determining spring coefficients in order to obtain a desired behavior is rather unwieldy for such meshes [VG98], although good results have been reported for some applications [BC00, LSH07].

Methods based on continuum mechanics offer more accuracy and flexibility in this context. Unlike direct methods, there is a clear separation between the physical model, described mathematically in terms of partial differential equations (PDEs), and its spatial discretization with finite differences or finite elements. Terzopoulos et al. [TPBF87] introduced this rigorous approach to computer graphics, using principles from differential geometry and finite difference discretizations in order to simulate elastic curves, surfaces, and solids. A continuum mechanics treatment of thin flexible surfaces is in general based on thin plate or shell theory. While this is the approach favored in textile engineering (see, e.g., [Llo80, CCOS91, GLS95]), only few works have been reported in the context of computer animation [EDC96, GKS02, TWS06]. The vast majority builds on a splitting of membrane and bending behavior, treating only the former with continuum-based methods using nonlinear [WDGT01] or linear [EKS03] finite elements.

There also exists a number of works that take a middle route between pure continuum-based approaches and direct methods. Drawing on concepts from continuum mechanics, Volino et al. [VCMT95] compute in-plane forces on unstructured triangle meshes. Similarly, also Baraff and Witkin [BW98] take into account the material's orientation when computing stretch, shear, and even bend forces. Eischen and Bigliani [EB00] described a modified mass-spring system on

[2] see House and Breen [HB00] for a more extensive account of existing work, including procedural methods

a regular grid and obtained good agreement with finite element simulations on simple examples. Correspondences between particle systems and continuum mechanics were investigated by Etzmuß et al. [EGS03] for regular discretizations and by Delingette [Del08] in the context of unstructured triangle meshes. Close to continuum mechanics are also the approaches described by Volino et al. based on particle systems with linear [VMT05a] and nonlinear [VMTF09] in-plane deformation measures.

Most approaches treat bending apart from in-plane deformations. Provot [Pro95] describes a simplistic approach which uses interleaved springs to model bending resistance on regular grids. Using the same discretization, Choi and Ko [CK02] present an extension to this model, assuming that cloth buckles immediately at the onset of compression. A method applicable to unstructured triangle meshes was described by Volino et al. [VCMT95], who compute bending forces using an approximate curvature measure defined on pairs of edge-adjacent triangles. A similar approach was used by Baraff and Witkin [BW98]. A more rigorous formulation that accounts for a clean separation of membrane and bending contributions are the discrete shell energy by Grinspun et al. [GHDS03] and the equivalent force formulation of Bridson et al. [BMF03]. While methods based on discrete curvature measures offer a high level of accuracy, they also involve considerably more intensive computations than simpler models. Subsequent work therefore aimed at improving performance, making restrictive assumptions on deformation in order to derive bending forces that are linear [BWH+06, VMT06b] or quadratic [GGWZ07] functions of positions. While the aforementioned bending models are based on standard discretizations, other methods have considered point-based [WSG05] and discontinuous [KMBG09, KMB+09] discretizations as well.

An important aspect of physical cloth simulation is the way in which the complex, nonlinear material behavior of textiles is modeled or approximated. While earlier methods used quite elaborate models based on measured data [BHW94, EWS96, EDC96], many subsequent works have turned to linear relationships between force and deformation for reasons of simplicity and efficiency. In particular, Baraff and Witkin [BW98] suggest to use '*a very stiff stretch force*' in order to prevent large in-plane deformations. But besides the fact that this leads to a considerable increase in artificial damping, it is also not sufficient for capturing the diversity of woven cloth materials, which do not stretch excessively but oppose only weak resistance to small deformations. This behavior is better approximated with a combination of weak elastic forces and constraints that prevent large stretch deformations. This approach was first employed by Provot [Pro95], who used weak linear springs with iterative length corrections for edges exceeding the deformation limit. The latter became known as strain limiting and was adapted and extended to a velocity-formulation [BFA02], constraints on triangle meshes [Tsi06], and general position-based constraints [MHHR06, Mül08]. While these methods handle constraints in an iterative way, direct schemes that enforce all constraints simultaneously have been proposed as well. Hong et al. [HCJ+05] describe a method to enforce linearized length constraints on a subset of a mesh's edges. Goldenthal et al. [GHF+07] presented an efficient constraint projection method for simulating inextensible cloth using quad-dominant meshes. An exten-

sion to triangle meshes was described by English and Bridson [EB08].

In order to resolve high frequency details such as small folds and wrinkles a high mesh resolution is required. However, these features are typically sparse and change locations over time, which motivates research into adaptive discretizations. There are not many methods to pursue adaptive strategies for cloth simulation, an exception being the work of Volkov and Li [VL03, LV05], who use discrete curvature measures to trigger dynamic refinement operations on triangular meshes. Another method in this direction is due to Wu et al. [WDGT01], who combine dynamic progressive meshes and nonlinear triangle finite elements for adaptive simulation of elastic membranes and other deformable bodies. Related is also the adaptivity framework described by Grinspun et al. [GKS02], which refines the basis functions of a given finite element space instead of geometry. This method makes no specific assumption on the nature of the elements and can therefore also be applied to thin-shell simulation.

The methods discussed above are mostly intended for simulating woven cloth or general textile materials. However, there are also approaches that address the particular structure of knitwear with particle systems [EMS00] or even at the yarn level [KJM08].

Numerical Time Integration There are numerous approaches to physically-based cloth simulation and the ways in which internal forces are computed differ considerably. However, all of them eventually lead to a set of ordinary differential equations with respect to time, which have to be solved by means of numerical time integration. Terzopoulos et al. [TPBF87] already used implicit schemes for this purpose and investigated both direct and iterative methods for solving the arising linear system. Subsequent work, however, turned to simpler explicit integration schemes such as the second order midpoint method used by Volino et al. [VCMT95] or the fourth order Runge-Kutta scheme employed by Eberhardt et al. [EWS96].

A paradigm shift was initiated when Baraff and Witkin [BW98] demonstrated the performance and stability of implicit methods on complex cloth animations. They used the semi-implicit[3] Euler scheme and solved the arising linear system with a modified conjugate gradients method. This approach was adopted in numerous subsequent work and is still widespread today, although later research identified considerable numerical damping as an inherent drawback. Hauth et al. [HE01a] showed that this unwanted effect can be alleviated by solving the full nonlinear equations and by resorting to higher-order methods such as the second order BDF-2[4] scheme. Subsequent work by Choi and Ko [CK02] reported good stability and low numerical dissipation for a semi-implicit version of the BDF-2 scheme, which was recently also used by English and Bridson [EB08] for animating inextensible triangle meshes. Another line of work [EEH00, BFA02, HES03, BA04] investigated IMEX methods, which treat only the stiff part of the equations with an implicit method while the remaining component is integrated explicitly. Related to this are also certain variants of the Newmark scheme, which have been used by Bridson et al. [BMF03] and Grinspun et al. [GHDS03]. Volino et al. [VMT05b]

[3] Semi- or linearly implicit methods solve only the linearized version of an actually nonlinear problem, see also [HW02].

[4] BDF denotes the family of backward differential formulas, see [HW02].

revisited the implicit midpoint rule and suggested a careful adjustment of damping parameters in order to improve stability. Addressing computer animation in general, Kharevych et al. [KYT+06] considered variational integrators, which are renowned for their momentum and energy conservation over long run times.

1.3.2 Collision Handling

Cloth animations generally involve frequent and complex collisions, which have to be detected and handled in order to prevent unphysical intersections. Designing robust and efficient methods for collision handling is a challenging problem and the number of existing approaches is large. We can only discuss a subset here and refer to Lin and Gottschalk [LG98] and Teschner et al. [TKZ+04] for detailed overviews.

Collision Detection The problem of collision detection has been studied intensively in the context of computer animation. Many efficient methods have been presented for special cases, including collisions between convex (rigid) polyhedra [GJK88, LC91, Mir98], parametric surfaces [VHBZ90], or reduced deformable models [JP04]. Cloth simulation is mainly concerned with deforming triangle meshes and in this case, collision detection amounts to determining all pairs of triangles that are in close proximity or intersect. Since exhaustive testing is prohibitively expensive, acceleration techniques such as bounding volume hierarchies are required in order to reduce the number of tests. Existing approaches differ primarily in the type of bounding volume that is used and, among others, methods based on axis-aligned bounding boxes [vdB98], sphere-trees [PG95, Hub96], oriented bounding boxes [GLM96], and discrete oriented polytopes (k-DOPs) [KHM+98, MKE03] have been described. A robust algorithm also has to account for collisions that occur between the discrete instants dictated by time integration. For this purpose, Moore and Wilhelms [MW88] used swept bounding volumes that enclose a triangle's position at the beginning and at the end of a given interval. This idea of continuous or geometric collision detection is readily extended to bounding volume hierarchies [Pro97, BFA02].

Bounding volume hierarchies can also be used to detect self-collisions, but the standard approach is inefficient in this case. Bounding volumes of adjacent regions (and triangles) of the surface necessarily overlap such that a large number of tests has to be performed even if there are no self collisions. Volino and Magnenat-Thalmann [VMT94] combine adjacency information between surface regions and curvature measures in order to skip tests for flat areas. Similar in spirit are the normal cones used by Provot [Pro97] to determine whether a given surface patch can contain self intersections. An extension of normal cones for continuous collision detection has been presented by Tang et al. [TCYM09].

The hierarchies of deforming objects have to be updated regularly in order to reflect changes in geometry. An optimal fit of bounding volumes is obtained when rebuilding the hierarchy from scratch in each time step. This is, however, quite costly and merely refitting the existing bounding volumes while maintaining the same structure is typically more efficient [LAM01, MKE03]. In order to further accelerate the update process, Mezger et al. [MKE03] propose a method

that exploits temporal coherence to update only those parts of the hierarchy that have changed significantly. A similar goal is pursued by Zachmann and Weller [ZW06], who use kinetic data structures in order to minimize hierarchy updates. The work of Otaduy et al. [OCSG07] addresses the problem of topology changes during simulation. By combining dynamic refitting and restructuring techniques, efficiency is maintained even in case of fracturing and tearing.

There also exist several methods that aim at saving time during hierarchy traversal. Li and Chen [LC98] describe a method for incremental collision detection that uses so called separation lists. For a given pair of hierarchies, a separation list keeps track of the bounding volume pairs that separate overlapping from non-overlapping parts of the corresponding trees. Instead of starting the hierarchy traversal at the root nodes in each time step, the detection process is warm-started with the nodes from the separation list. While this approach can greatly reduce the number of overlap tests during detection, it introduces overhead in cases when formerly close surface parts are separating again. The list has to be rebuilt in such cases in order to maintain efficiency, but deciding when to do this is typically difficult. Another line of work has considered stochastics in order to balance robustness against performance [KZ03, KNF04]. While these methods may be interesting for time critical applications, their non-conservative nature seems inappropriate for general computer animation.

A collision query (with bounding volume hierarchies) results in a set of potentially interfering triangle pairs, which need further processing before collision responses can be computed. To this end, triangle pairs are first decomposed into sets of vertex-triangle and edge-edge pairs. Geometric distance tests are then performed to determine the primitive pairs for which a collision response needs to be generated. If the corresponding triangle pair results from a continuous collision detection step, the exact time of impact (if any) between the primitives has to be determined. For vertex-triangle pairs, Moore and Wilhelms [MW88] cast this problem as a fifth order polynomial. Provot [Pro97] reduced the problem to finding the roots of a third order polynomial and generalized the formulation to accommodate edge-edge collisions. Issues of robustness have been addressed in [BFA02]. The involved computations are quite expensive and much time can be saved by minimizing the number of pairs for which exact tests have to be performed. In order to avoid redundant primitive tests, Wong and Baciu [WB06] describe a marking scheme which assigns the vertices and edges of a given mesh to its triangles. In this way, each primitive pair is covered by exactly one triangle pair, thus eliminating duplicate tests. A similar method was described by Curtis et al. [CTM08]. The primitive bounding boxes described by Hutter and Fuhrmann [HF07] are a complementary approach that can be used to eliminate non-interfering primitive pairs before performing more expensive distance computations.

Bounding volume hierarchies are presumably the most widely used approach for collision detection in cloth simulation. However, there are also alternatives, such as methods based on spatial subdivision. In the simplest case, a uniform spatial grid is used to subdivide the scene into regular cells, to which the primitives of deforming and rigid objects are registered. If two primitives are assigned to the same cell, they are considered as collision candidates and further tests are performed. This approach was first used by Turk [Tur89] in the context of molec-

ular dynamics and extensions for cloth simulation have been presented in [ZY00]. Storing the entire grid leads to excessive memory requirements, which is why sparse grids that only store occupied regions have been proposed. Teschner et al. [THM+03] describe a hash function that maps identifiers of occupied cells to their storage location. However, a drawback of this method is that different cells can be mapped to the same location. A hash function that guarantees injective cell mapping was described by Lefebvre and Hoppe [LH06].

Another alternative is to use distance fields in order to detect collisions between deformable surfaces and complex (static) rigid objects. A continuous distance field is a scalar function that assigns each point in space its distance to the closest point on a given surface. Discrete distance fields sample this function on uniform [OF02] or adaptive grids [FPRJ00]. Distance fields allow fast proximity queries for vertices but further treatment is necessary in order to avoid edge intersections. Another drawback is the high computation time required for constructing the distance fields. Consequently, they are either limited to static rigid objects or have to be precomputed for animated models [BMF03].

Another line of work has explored image-space techniques, which exploit the rasterization power of graphics hardware to accelerate collision detection. One of the first methods aimed at cloth animation is the the work of Vassilev et al. [VSC01], who project the scene along various directions and use the resulting depth maps in order to find collisions. While this method was limited to collisions between cloth and convex rigid bodies, generalizations to self-collisions and arbitrarily shaped objects have been described by Baciu and Wong [BW04] and Heidelberger et al. [HTG04]. Related is also the work of Govindaraju et al. [GKJ+05], who use chromatic mesh decomposition and GPU computations for efficient culling among non-adjacent triangles. Subsequent work from the same group suggested discrete Voronoi diagrams [SGG+06], which can account for topology changes and additionally support local distance and penetration queries. A method that combines image-based collision detection and response was recently described by Faure et al. [FBAF08].

Collision Response Once all primitive pairs that need treatment are determined, appropriate collision responses have to be generated in order to prevent intersections. We can essentially distinguish between three types of responses, which are forces, constraints, and impulses. However, some approaches also use combinations thereof.

Terzopoulos et al. [TPBF87] derived repelling forces from the gradient of a potential field defined around rigid objects. Moore and Wilhelms [MW88] temporarily insert elastic springs that push too close points apart. Carignan et al. [CYTT92] use a modified formulation that results in a completely inelastic collision response.

Baraff and Witkin [BW98] combine damped spring forces for self collisions and constraints for object-cloth collisions. Both are integrated in the implicit time stepping scheme: spring forces and their derivatives are added to the equations of the implicit solver while constraints are enforced by appropriate filtering during the conjugate gradients iterations.

Provot [Pro97] handles collisions with momentum preserving impulses that,

for a given vertex-triangle or edge-edge pair, prevent further approach. His formulation covers elastic as well as inelastic collisions and also accounts for simple Coulomb friction. This model was also adopted in [BFA02] and extended by repelling impulses that push primitives in too close proximity apart.

The above described methods span the diversity of basic collision response, but on their own they are not sufficiently robust to reliably prevent intersections. The impulse-based response, for instance, resolves collisions for a given primitive pair but intersections can still occur for primitives involved in multiple collisions. Provot [Pro97] proposed an iterative strategy to solve this problem. After collisions have been treated with stopping impulses an additional detection step is performed. If there are remaining intersections, all vertices participating in a given set of multiple collisions are gathered into an impact zone, which is then moved rigidly. This step is repeated, leading to growing and merging impact zones, until finally all intersections are resolved. This approach was adopted by Bridson et al. [BFA02] who extended Provot's method to a three stage framework for robust, iterative collision handling. This method consist of (*1*) a single pass of simple collision detection and response via stopping and repelling impulses, (*2*) possibly multiple iterations of continuous collision detection and impulse-based response, and (*3*) treatment with rigid impact zones, possibly requiring multiple iterations. This framework appeals through efficiency, versatility and robustness, which is why it has found widespread use in research as well as industry.

All problems are, however, not solved, which is testified by various subsequent works and improvements. For example, Harmon et al. [HVTG08] resolve the problem that rigid impact zones eliminate not only those velocity components that cause collisions, but also tangential motion. Sifakis et al. [ES08] propose another replacement for the second stage (and indirectly also the third stage) of [BFA02], which uses the metaphor of an incompressible fluid in order to couple primitive responses globally, leading to a system of nonlinear equations.

A limitation of impulse-based collision handling is the fact that there is no coupling between physics and collisions. Consequently, rapidly moving characters or high-velocity self collisions can induce significant amounts of deformation, which destabilize the simulation. The approach described by Otaduy et al. [OTSG09] addresses this problem by directly coupling the time integration of physics and collision response within a constrained dynamics formulation.

Even with robust collision handling methods, there can still be extreme situations in which not all collisions are resolved and this requires additional treatment. Resolving intersections *a posteriori* has been an early concern in cloth simulation [VCMT95], but more robust methods have been presented recently for this purpose. Baraff et al. [BWK03] proposed a method that uses global intersection analysis to recover a valid configuration for meshes tangled due to self-intersecting character motion. The approach described by Volino and Magnenat-Thalmann [VMT06a] minimizes the length of intersection contours, which leads to further improved robustness and, in particular, allows the resolution of intersections involving mesh boundaries.

Collision response is necessary to prevent intersections but it is also the context for dealing with frictional contact. While earlier approaches used spring forces for this purpose, more recent methods rely mostly on impulse-based implementations

of isotropic [Pro95, BFA02, SSIF08] or anisotropic [PKST08] Coulomb friction. A rigorous impulse-based treatment of frictional contact can also be found in the work of Cirak and West [CW05], who use velocity decompositions in order to compute accurate, momentum-conserving contact responses.

Parallel Methods Despite significant advances over the past two decades, physically-based cloth simulation is still very time consuming. This motivates research into parallel methods and several approaches have been presented, targeting distributed-memory [ZFV04, KB04, SSIF08] and shared-memory [RRZ00, LGPT01, GRR+05] architectures. But in spite of its importance for general animations, only Romero et al. [RRZ00] explicitly addressed the problem of collision detection. Parallel collision detection has previously been studied in the context of various applications from the engineering domain, e.g., for crash impact simulation [BASH00] or projectile penetration [Kar03]. Most of these applications are concerned with volumetric solids simulations in which contact is restricted to a small set of surface elements. This is considerably different from cloth simulations, where every element is a surface element that can be involved in multiple external and internal collisions. Closer to the needs of computer graphics is the parallel method by Lawlor et al. [LK02], which is based on spatial subdivision with a sparse uniform grid. Though promising results were reported for this approach, its parallel efficiency is critically influenced by the voxel size and finding a value that leads to balanced workloads while keeping parallel overhead low is difficult.

The first parallel method explicitly tailored for cloth simulation is due to Romero et al. [RRZ00] who propose an approach based on bounding volume hierarchies and separation lists. However, only the separation lists are processed in parallel such that the hierarchy test is bound to become a sequential bottleneck, thus posing severe limits on scalability.

1.3.3 Rigid Body Interaction

Volumetric objects are pervasive in virtual environments. If deformations are negligible or unimportant, it is wasteful to use discrete elastic models with many degrees of freedom. A more efficient approach is to approximate such objects as ideal rigid bodies, which can translate and rotate, but not deform. The spatial configuration of a rigid body is described by the position of its center of mass and its current orientation, which amounts to six degrees of freedom. The governing principles of rigid body dynamics are well known an detailed expositions can be found in most standard textbooks on classical mechanics, such as [Arn97]. A good overview and practical introduction to the field can be found in the course notes by Baraff [Bar01]. In the following, we will briefly comment on the major challenges in rigid body simulation and summarize some of the developments that emerged in the context of computer graphics.

Collisions, Contact, and Friction The dynamics of rigid bodies are governed by a set of ordinary differential equations, consisting of a linear part according to Newton's second law of motion and a rotational part described by Euler's equa-

tions [Arn97]. Integrating these equations for isolated rigid bodies is not particularly difficult, but simulating multiple bodies with collisions and contact is a challenging problem. The first methods in graphics that account for such interactions are due to Moore and Wilhelms [MW88] and Hahn [Hah88]. Collisions and contacts between pairs of objects were treated sequentially and in isolation, using analytically computed impulses or penalty forces to model the response. But despite later improvements by Mirtich and Canny [MC95], a lack of accuracy with respect to frictional contact remained a major drawback.

Another class of approaches are constraint-based methods, in which the simultaneous computation of all contact responses is cast as a global optimization problem [Cot77, Löt84]. This involves formulating a set of (inequality) constraints and (complementarity) conditions on forces and accelerations at contact points. In computer graphics, the first approaches in this direction are due to Baraff, who initially considered only frictionless contact [Bar89a]. Subsequent work by the same author [Bar91, Bar94] addressed frictional contact but in this case, the formulation in terms of accelerations and forces cannot guarantee the existence of solutions. However, this issue can be resolved by formulating contact processes in terms of velocities and impulses [ST96, AP97]. See also Stewart [Ste00] for further details on constrained-based modeling of frictional contact and a more extensive account of existing methods.

The approaches mentioned so far (may) perform well for small problems but become inefficient when applied to systems with large numbers of rigid bodies. One line of subsequent research improved on this by proposing, for instance, simplified physics [Mil96], event-based time stepping [Mir00] or efficient techniques based on local [KEP05] or global [MS01] optimization. The impulse-based method by Guendelman et al. [GBF03] aims at simulating complex stacking effects such as large piles of nonconvex objects. Collisions and contact are treated in two separate passes and convergence of the latter is accelerated using shock propagation. An extended method with improved computational efficiency and better accuracy was described by Erleben [Erl07]. Noteworthy in this context is also the work by Kaufman et al. [KSJP08] who addressed the simulation of rigid and deformable bodies in frictional contact.

Articulation, Control, and Coupling Another early emphasis of rigid body simulation in computer graphics has been the animation of articulated figures, in particular human locomotion [AG85]. A variety of constraints can be used to model articulations (see, e.g., [IC87]) and there are numerous methods for simulating the resulting systems, including those based on reduced coordinates [Fea87], Lagrange multipliers [BB88b, Bar96] or impulses [WTF06].

With conventional methods, a user can only influence initial values and simulation parameters, but practical applications typically require additional control over animations. As a straightforward approach, keyframe constraints can be integrated into dynamic simulations using inverse kinematics [IC87], but the arising artificial forces can lead to implausible motion. A more rigorous approach is to use spacetime optimization [WK88, Coh92]. The latter computes control forces and motion trajectories that satisfy user-specified constraints while minimizing an objective function, such as the work done by the muscles of a humanoid fig-

ure. Other approaches do not rely on optimization but use control algorithms to compute feedback forces that are directly integrated with the simulation [RH91]. While these methods are primarily aimed at articulated rigid bodies with muscle-like actuators, other approaches have investigated the control of animations in the absence of self-propelling forces. In this case, a set of parameters (e.g., initial values, material properties, or collision normals) is sought such that the resulting simulation matches the desired behavior [BB88b, CF00]. The latter can also be specified interactively via position, orientation and velocity constraints [PSE+00] or by sketching trajectories [PSE03]. Another approach is to use time-reversed simulation [TJ08] to compute directed physical motion by stepping a system backward in time from a given target configuration.

Finally, complex animations may also involve interactions of rigid bodies with other non-rigid systems, for which different levels of coupling can be pursued [OZH00]. A large number of works has addressed the coupling of rigid bodies and fluids (including [YOH00, CMT04, KFCO06, RMSG+08]), but also the interaction with deformable models has been investigated [SSIF07, SSF08].

Chapter 2

Physical Cloth Simulation

This chapter describes the computational framework for physically-based cloth simulation which serves as the basis for the developments presented in Chapters 3 to 5. A cloth simulator consists of a number of different components, each requiring careful design choices. The most fundamental question is how to best translate the mechanical behavior of fabrics into a discrete model that is amenable to efficient computer implementation. To answer this, we must define the macroscopic properties that have to be reflected by the discrete approximation.

Textiles are thin flexible materials that can deform into complex shapes with characteristic folding and buckling patterns as illustrated in Fig. 2[1]. The visual appearance of cloth is usually dominated by large bending deformations, but the way in which folds and wrinkles form also depends strongly on in-plane properties. The latter are often direction-dependent and, due to the yarn structure, especially woven fabrics typically exhibit a nonlinear and anisotropic stretch resistance.

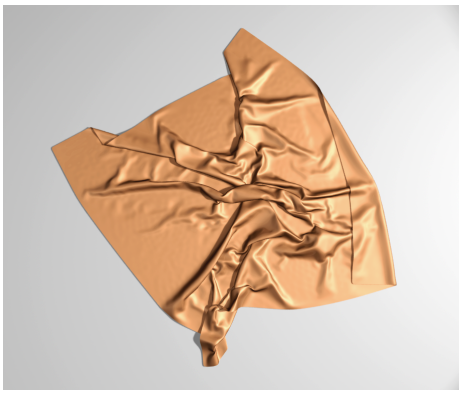


Figure 2.1: Typical folding and buckling patterns of cloth.

Mass-spring systems are well liked for their simplicity, but while good results *can* be obtained, their inherent dependence on discretization is a severe practical limitation. By contrast, continuum mechanics offers a sound basis for a discrete model which is flexible with respect to discretizations and allows accurate control over material behavior. Thin shell theory [SF89, WT03] provides the appropriate context for bending-dominated problems, but the high smoothness requirements and the associated computational costs have so far prevented a widespread acceptance for cloth simulation. A widely used and computationally more attractive alternative is to neglect the coupling between membrane[2] and bending mechanics and to treat each component separately. As for the bending part, a number of discrete models that offer good accuracy but avoid the complexity of thin-shell approaches have re-

[1] frame selected from Example 3.4 of Chapter 3

[2] We will collectively refer to stretching and shearing as membrane deformation in the remainder.

cently been presented. The membrane part can be modeled accurately and at comparatively low computational costs using continuum mechanics and finite elements. Combining a discrete bending model with a continuum-based membrane model thus constitutes a good balance between accuracy and efficiency, which is why we adopt this approach in this work. The resulting mechanical model as well as the remaining components of the cloth simulation framework are described in the remainder of this chapter, which is structured as follows.

Overview Being the basis of the membrane model, we start by summarizing the relevant concepts of continuum mechanics in Sec. 2.1. This includes measures for strain and stress as well as their interrelating material laws, which are chosen in order to accommodate finite deformations and anisotropic material behavior. Discrete internal forces are derived in Sec. 2.2 and, in particular, a finite element method based on linear triangles is proposed for the membrane part. Having established the discrete equations of motion, we turn to their numerical solution and discuss several time integration schemes in Sec. 2.3. The cloth simulation framework is completed by a robust collision handling method, which is described in Sec. 2.4. The chapter concludes with a summary of the presented material in Sec. 2.5.

2.1 Continuum Mechanics

In the idealized view of continuum mechanics, it is assumed that all quantities of interest are continuously distributed over the problem domain under consideration. Although textiles typically possess a heterogeneous structure, the continuum approximation is valid for most fabric materials if only macroscopic behavior is considered.

We start by introducing the concepts of deformation and stress in the three-dimensional setting and subsequently specialize to two dimensions in the context of material laws. For conciseness, we only address parts that are directly relevant to this work and refer to the textbooks by Bathe [Bat96] and Bonet and Wood [BW97] for a comprehensive introduction to continuum mechanics.

2.1.1 Deformation

Let $\mathcal{B}$ denote a deformable body occupying a spatial region $\bar{\Omega} \in \mathbb{R}^3$ and let its motion be described by a time-dependent function $\varphi : \bar{\Omega} \times [0, \infty) \to \Omega \subset \mathbb{R}^3$. The mapping φ transforms positions of material particles in the initial configuration to positions in the current configuration as

$$\mathbf{x}_i(t) = \varphi(\mathbf{x}_i(0), t) \, . \tag{2.1}$$

φ is commonly referred to as the *configuration mapping*. For brevity, we write

$$\mathbf{x}_i = \mathbf{x}_i(t) \, , \qquad \bar{\mathbf{x}}_i = \mathbf{x}_i(0) \quad \text{and} \qquad \varphi(\cdot) = \varphi(\cdot, t) \, . \tag{2.2}$$

A central aspect in the simulation of deformable media is how to measure deformation. Consider two neighboring material particles $\bar{\mathbf{x}}_1$ and $\bar{\mathbf{x}}_2$ such that

$d\bar{\mathbf{x}}_{12} = \bar{\mathbf{x}}_2 - \bar{\mathbf{x}}_1$ is of infinitesimal length. In the current configuration, we have

$$\mathbf{x}_1 = \varphi(\bar{\mathbf{x}}_1) \,, \quad \mathbf{x}_2 = \varphi(\bar{\mathbf{x}}_2) = \varphi(\bar{\mathbf{x}}_1 + d\bar{\mathbf{x}}_{12}) \quad \text{and} \quad d\mathbf{x}_{12} = \mathbf{x}_2 - \mathbf{x}_1 \,. \tag{2.3}$$

Using a Taylor series expansion and truncating after linear terms, we can write

$$d\mathbf{x}_{12} = \varphi(\bar{\mathbf{x}}_1 + d\bar{\mathbf{x}}_{12}) - \varphi(\bar{\mathbf{x}}_1) \approx \varphi(\bar{\mathbf{x}}_1) + \nabla\varphi \cdot d\bar{\mathbf{x}}_{12} - \varphi(\bar{\mathbf{x}}_1) = \nabla\varphi \cdot d\bar{\mathbf{x}}_{12} \,. \tag{2.4}$$

The gradient of the deformation mapping, or *deformation gradient*, is a fundamental quantity and we define

$$\mathbf{F} = \nabla\varphi = \frac{\partial\varphi}{\partial\bar{\mathbf{x}}} \,. \tag{2.5}$$

Visually, $\mathbf{F}$ maps vectors in the initial configuration to their deformed counterparts as $d\mathbf{x} = \mathbf{F}d\bar{\mathbf{x}}$. Closely related to this quantity, we can now define a deformation measure as follows. Let $\bar{\mathbf{x}}_3$ denote the position of an additional material particle in close vicinity to $\bar{\mathbf{x}}_1$ and define $d\bar{\mathbf{x}}_{13} = \bar{\mathbf{x}}_3 - \bar{\mathbf{x}}_1$. A suitable deformation measure should capture pure stretching but also angular changes between pairs of vectors. This requirement is satisfied by the difference in dot product,

$$d\mathbf{x}_{12}^t d\mathbf{x}_{13} - d\bar{\mathbf{x}}_{12}^t d\bar{\mathbf{x}}_{13} = d\bar{\mathbf{x}}_{12}^t(\mathbf{F}^t\mathbf{F} - \mathbf{I})d\bar{\mathbf{x}}_{13} \,. \tag{2.6}$$

The middle term is independent of the choice of material vectors and is therefore a general description of deformation, which leads to the *Green strain* tensor

$$\mathbf{E} = \frac{1}{2}(\mathbf{F}^t\mathbf{F} - \mathbf{I}) \,. \tag{2.7}$$

The Green strain is a nonlinear tensor, which is clearly symmetric and additionally invariant under rigid rotations. To see this, note that every deformation $\mathbf{F}$ can be decomposed[3] into a rotational part $\mathbf{R}$ and a pure stretch tensor $\mathbf{U}$ as $\mathbf{F} = \mathbf{R}\mathbf{U}$. Introducing the right Cauchy-Green tensor $\mathbf{C}$ as

$$\mathbf{C} = \mathbf{F}^t\mathbf{F} = \mathbf{U}^t\mathbf{R}^t\mathbf{R}\mathbf{U} = \mathbf{U}^2 \,, \tag{2.8}$$

the rotation invariance of $\mathbf{E} = \frac{1}{2}(\mathbf{C} - \mathbf{I})$ becomes evident.

If only small displacements are expected, it is often sufficient to approximate (2.7) by a linearized strain measure. Introducing the displacement field $\mathbf{u}$ as

$$\varphi(\bar{\mathbf{x}}) = \bar{\mathbf{x}} + \mathbf{u} \,, \qquad \mathbf{u} = \mathbf{x} - \bar{\mathbf{x}} \,, \tag{2.9}$$

the deformation gradient can be written as

$$\mathbf{F} = \mathbf{I} + \nabla\mathbf{u} \,, \tag{2.10}$$

where $\mathbf{I} \in \mathbb{R}^{3\times3}$ denotes the identity matrix. This allows us to express $\mathbf{E}$ in terms of displacements as

$$\mathbf{E} = \frac{1}{2}(\nabla\mathbf{u}^t + \nabla\mathbf{u} + \nabla\mathbf{u}^t\nabla\mathbf{u}) \,. \tag{2.11}$$

[3]Every real-valued, non-singular matrix can be expressed as the product of an orthogonal and a positive semi-definite matrix. This is decomposition is known as *polar decomposition* [GVL96].

The linearization of $\mathbf{E}$ amounts to dropping nonlinear terms from the above expression and we obtain the linear *Cauchy strain* tensor as

$$\varepsilon = \frac{1}{2}(\nabla \mathbf{u}^t + \nabla \mathbf{u}) \; . \tag{2.12}$$

This tensor enjoys widespread use in computer graphics since its linear nature promises good computational efficiency. A fundamental shortcoming is, however, the fact that ε is not invariant under rotations. This is unfortunate for the simulation of deformable surfaces, which typically undergo large rotational motion. In order to obtain a linear and yet rotationally invariant measure, several methods have been proposed to extract the rotational component of the displacement field prior to strain computation[4]. Supposing that the rotation field $\mathbf{R}$ can be computed throughout the domain, the *co-rotational strain* tensor can be defined as

$$\varepsilon_{cr}(\mathbf{R}\mathbf{u}) = \varepsilon(\mathbf{R}^t\mathbf{R}\mathbf{u}) = \varepsilon(\mathbf{u}) \; . \tag{2.13}$$

In order to accurately account for finite rotations, we build the membrane model on the basis of the Green strain, resulting in a geometrically nonlinear approach. However, the co-rotational strain tensor will be used in the context of continuum-based deformation limiting, which is described in Chapter 3.

2.1.2 Stress and Equilibrium

Traction loads on the boundary of an elastic solid as well as body forces, e.g., due to gravity provoke balancing internal forces. The distribution of internal forces is described by the *Cauchy stress* tensor σ. This symmetric tensor maps normal directions $\mathbf{n}$ to traction force densities $\mathbf{t}$ acting on the corresponding material plane,

$$\mathbf{t}(\mathbf{x}, \mathbf{n}) = \sigma(\mathbf{x}) \cdot \mathbf{n} \; , \qquad \mathbf{x} \in \Omega \; . \tag{2.14}$$

The relation between internal stress and externally applied loads is described by the point-wise equilibrium conditions of continuum mechanics, which are[5]

$$\begin{aligned} \operatorname{div} \sigma(\mathbf{x}) + \mathbf{b}(\mathbf{x}) &= 0 \; , \qquad \mathbf{x} \in \Omega \; , & (2.15) \\ \sigma(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}) - \mathbf{s}(\mathbf{x}) &= 0 \; , \qquad \mathbf{x} \in \partial\Omega \; . & (2.16) \end{aligned}$$

In these expressions, $\mathbf{b}$ and $\mathbf{s}$ are externally applied body force and surface traction densities and $\mathbf{n}$ denotes the outward normal on the boundary of the solid $\partial\Omega$.

The Cauchy stress tensor is a *spatial* quantity that maps between vectors in the deformed configuration. However, it is computationally more convenient to use only quantities that refer to the initial configuration. To this end, we introduce the second Piola-Kirchhoff stress tensor $\mathbf{S}$, which is a symmetric *material* quantity that maps unit directions in the initial configuration to force densities in the same space. $\mathbf{S}$ is related to the Cauchy stress σ via the so called Piola transformation,

$$\mathbf{S} = J\mathbf{F}^{-1}\sigma\mathbf{F}^{-t} \; , \qquad \text{where} \qquad J = \det \mathbf{F} \; . \tag{2.17}$$

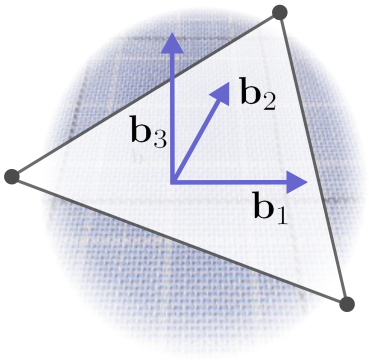
[4] for graphics-related works see, e.g., [MDM+02, EKS03, HS04, TWS06, MTPS08]

[5] see Appendix A.1

Stress can develop in a deformable body as an elastic response to deformation, but also material viscosity, plasticity or thermal effects can be sources of internal forces. We will consider only elastic and viscous stress contributions in this work and describe a corresponding material law in the following section.

2.1.3 Material Laws

We have so far only considered the general case of three-dimensional elasticity, for which strain and stress tensors can be expressed as symmetric (3×3) matrices. For thin membranes, however, a reduction to two dimensions can be achieved by introducing appropriate kinematic assumptions.



To this end, we assume that $\mathbf{E}$ is expressed with respect to a local coordinate system with mutually orthogonal basis vectors $[\mathbf{b}_1\ \mathbf{b}_2\ \mathbf{b}_3]$ such that $\mathbf{b}_1$ and $\mathbf{b}_2$ are aligned with the material's main directions and $\mathbf{b}_3$ corresponds to the thickness direction as illustrated in the Figure to the right. In particular, we define $\mathbf{b}_1$ as the weft and $\mathbf{b}_2$ as the warp yarn direction for woven fabrics. In this setting, the components of $\mathbf{E}$ can be interpreted as follows: $\mathbf{E}_{11}$ and $\mathbf{E}_{22}$ describe the stretch deformation in weft and warp direction, while $\mathbf{E}_{12}$ corresponds to in-plane shearing. Furthermore, $\mathbf{E}_{i3}$ for $i = \{1,2\}$ and $\mathbf{E}_{33}$ describe shearing and stretching in the thickness direction, respectively.

Cloth is typically thin and it is therefore reasonable to assume that stresses in the thickness directions are negligible compared to the in-plane components [ZT00a]. In terms of the second Piola-Kirchhoff tensor $\mathbf{S}$, this condition of *plane stress* is expressed as

$$\mathbf{S}_{i3} = \mathbf{S}_{3i} = 0 \qquad \text{for} \quad i \in \{1,2,3\}\ .$$

We will additionally assume that there are no shear deformations in the thickness direction[6],

$$\mathbf{E}_{i3} = \mathbf{E}_{3i} = 0 \qquad \text{for} \quad i \in \{1,2\}\ .$$

Note that the stretch deformation $\mathbf{E}_{33}$ in thickness direction can be nonzero, but it is only a function of the in-plane stresses and can therefore be eliminated. This effectively reduces the analysis to two dimensions, leaving the in-plane components of stress and strain as the primary problem variables. Having restricted the computations in this way, we can now turn to the actual relation between strain and stress.

For a purely elastic[7] material, the stress is only a function of the current deformation and there exists an elastic potential Ψ such that

$$\mathbf{S}(\mathbf{C}) = 2\frac{\partial \Psi(\mathbf{C})}{\partial \mathbf{C}}\ . \tag{2.18}$$

[6] This is one of the Kirchhoff-Love assumptions for thin plates [WT03].

[7] The technically correct term is *hyperelastic*, which denotes an elastic material with path-independent behavior [BW97].

This formulation also emphasizes the fact that **S** is invariant under rigid rotations, since it depends only on the rotation-free tensor **C**, rather than directly on the deformation gradient **F**. The elastic stress is in general a nonlinear function of positions, but we can distinguish two types of nonlinearity in this context: a *geometric* nonlinearity that refers to the relation between strain and positions, and a *material* nonlinearity that refers to the relation between stress and strain. In order to deal with finite rotations in a stable and accurate manner, we choose the nonlinear Green strain instead of a linear deformation measure. We elaborate further on the advantages of this choice at the end of Sec. 2.2.1. Having settled for the Green strain, the simplest model is therefore geometrically nonlinear, but materially linear. If, additionally, a completely isotropic material behavior is assumed, we arrive at the well-known St.Venant-Kirchhoff model, whose elastic potential is given as[8]

$$\Psi_{iso} = \frac{1}{2}\lambda \mathrm{tr}(\mathbf{E})^2 + \mu(\mathbf{E} : \mathbf{E}) \ . \tag{2.19}$$

With λ and μ corresponding to the Lamé constants, this model is the direct extension of the linear isotropic material familiar from small strain analysis. Using Eq. (2.18), the second Piola-Kirchhoff stress follows as

$$\mathbf{S} = \lambda \mathrm{tr}(\mathbf{E})\mathbf{I} + 2\mu\mathbf{E} \ . \tag{2.20}$$

Isotropic models are only of limited use for textile materials. Orthotropic models are better suited for this case, since they allow material properties to be specified along two perpendicular directions[9], which fits well with the yarn structure of common woven fabrics. The elastic potential of the orthotropic St. Venant-Kirchhoff [10] variant is given as

$$\Psi_{ani}(\mathbf{C}) = \frac{1}{2}\sum_{i,j}^{2} a_{ij}\mathbf{E}_{ii}\mathbf{E}_{jj} + G(\mathbf{E}_{12}^2 + \mathbf{E}_{21}^2) \ , \tag{2.21}$$

where G is the shear modulus and the material constants a_{ij} are related to the familiar Young's moduli E_i and Poisson's ratios ν_{ij} as

$$a_{ii} = E_i \frac{1}{1-\nu_{ij}\nu_{ji}}, \quad a_{ij} = a_{ji} = E_i \frac{\nu_{ij}}{1-\nu_{ij}\nu_{ji}} \ . \tag{2.22}$$

We thus have four independent material coefficients relating strain to stress and using Eq. (2.21) in (2.18) we obtain

$$\mathbf{S}_{ii} = a_{ii}\mathbf{E}_{ii} + a_{ij}\mathbf{E}_{jj} \quad \text{and} \quad \mathbf{S}_{ij} = 2G\mathbf{E}_{ji} \ . \tag{2.23}$$

This expression describes a simple orthotropic stress tensor, which is – just like its isotropic counterpart – a linear function of the Green strain. This accounts for the direction-dependent nature of common textiles, but there is only a single

[8] The *trace* of a tensor is defined as the sum of its diagonal entries, $\mathrm{tr}(\mathbf{E}) = \sum_i \mathbf{E}_{ii}$. The *inner product* of two tensors is defined as $\mathbf{A} : \mathbf{B} = \mathrm{tr}(\mathbf{A}^t\mathbf{B})$.

[9] More formally, an orthotropic material is defined as having properties that are invariant under reflection with respect to two orthogonal planes, see [AG00].

[10] see e.g. [IA04]

set of material coefficients for the entire deformation range. This is, in general, not sufficient to model the characteristic stress-strain behavior of real fabrics. But instead of considering more advanced material models, we will extend this simple approach in a different way. This will be described in Chapter 3.

Textiles are not perfectly elastic but exhibit also viscous material properties. For applications such as virtual surgery, the viscous behavior is of central importance and justifies the use of advanced models [HGS03]. For general computer animation, however, a simpler approach based on the rate of strain tensor

$$\dot{\mathbf{E}} = \frac{1}{2}(\dot{\mathbf{F}}^t\mathbf{F} + \mathbf{F}^t\dot{\mathbf{F}}) \tag{2.24}$$

is typically sufficient [OH99, DDCB01]. In analogy to Eq. (2.20), we assume that the viscous stress $\mathbf{S}_v$ is a linear isotropic function of the strain rate,

$$\mathbf{S}_v = \lambda_d \mathrm{tr}(\dot{\mathbf{E}})\mathbf{I} + 2\mu_d\dot{\mathbf{E}}\,, \tag{2.25}$$

where λ_d and μ_d are corresponding damping coefficients. Since the strain rate $\dot{\mathbf{E}}$ is, just like the Green strain itself, invariant under rigid body motion[11], rotations are not damped using this formulation.

2.2 Discrete Internal Forces

Having derived expressions for membrane strains and stresses in the continuous setting, we can now proceed to their discretization. We assume that deformable surfaces are represented as triangle meshes with n_v vertices and n_f faces. Each vertex is associated with a nodal mass m_i as well as three-dimensional vectors $\mathbf{x}_i$ and $\mathbf{v}_i$ describing its current position and velocity.

2.2.1 Finite Element Membrane Forces

The term *Finite Elements* is, broadly speaking, used to designate a class of methods for the numerical treatment of partial differential equations. As one of many appreciable properties, finite elements can be applied to complex problems on arbitrary geometries. This is also attractive for cloth simulation, where fabric panels typically have curved boundaries and are therefore best modeled with unstructured triangle meshes.

The starting point for finite element discretizations is the weak form or variational formulation of the point-wise equilibrium conditions (2.15) and (2.16). Appendix A provides a detailed description of the necessary transformations, but for the sake of conciseness, we will restrict our considerations to a practical level in this chapter.

A finite element approach is largely characterized by the type of element used for discretization. Since we intend to limit computation costs as much as possible, we choose the simplest type of element, namely linear triangles. A linear triangle element $\mathcal{K}_e$ is defined by its domain $\bar{\Omega}_e \subset \bar{\Omega}$, its three nodal positions $\bar{\mathbf{x}}_i \in \mathbb{R}^2$,

[11] Since $\mathbf{F} = \mathbf{RU}$, we have $\dot{\mathbf{E}} = \frac{1}{2}(\dot{\mathbf{U}}^t\mathbf{U} + \mathbf{U}^t\dot{\mathbf{U}}) + \frac{1}{2}\mathbf{U}^t(\dot{\mathbf{R}}^t\mathbf{R} + \dot{\mathbf{R}}\mathbf{R}^t)\mathbf{U}$, but $\mathbf{R}^t\mathbf{R} = \mathbf{I}$ such that $\dot{\mathbf{R}}^t\mathbf{R} = -\mathbf{R}^t\dot{\mathbf{R}}$, which proves the invariance of $\dot{\mathbf{E}}$.

as well as its three linear *shape functions* N_i associated with the nodes. The shape functions are defined with respect to the two-dimensional rest state as

$$N_i : \bar{\Omega}_e \rightarrow [0,1] \ , \qquad N_i(\bar{\mathbf{x}}_j) = \delta_{ij} \ . \tag{2.26}$$

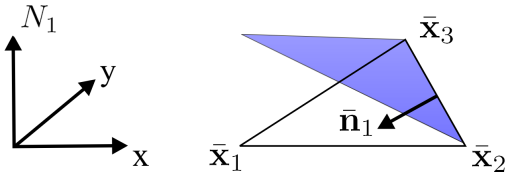


Figure 2.2: Linear shape function N_1.

Explicit expressions for the shape functions can readily be derived geometrically. To this end, we consider the shape function N_1 of node 1 as an example and note that $N_1(\bar{\mathbf{x}}) = 0$ on the edge $\bar{\mathbf{e}}_1 = \bar{\mathbf{x}}_3 - \bar{\mathbf{x}}_2$ while $N_1(\bar{\mathbf{x}}_1) = 1$ (see Fig. 2.2). Hence, the gradient of N_1 must be orthogonal to $\bar{\mathbf{e}}_1$ and its magnitude is determined as $\frac{1}{h_1}$, where h_1 is the height of node 1. Introducing the edge normal

$$\bar{\mathbf{n}}_1 = \frac{1}{||\bar{\mathbf{e}}_1||} \cdot \begin{bmatrix} -\bar{\mathbf{e}}_{1y} \\ \bar{\mathbf{e}}_{1x} \end{bmatrix} \tag{2.27}$$

we can write

$$N_1(x,y) = \frac{1}{h_1}(\begin{bmatrix} x \\ y \end{bmatrix} - \bar{\mathbf{x}}_2) \cdot \bar{\mathbf{n}}_1 \ , \tag{2.28}$$

noting that $\bar{\mathbf{x}}_2$ could be replaced by an arbitrary point on $\bar{\mathbf{e}}_1$. The shape functions themselves are not needed for computations, only their partial derivatives are required. The latter are obtained from Eq. (2.28) as

$$\frac{\partial N_1}{\partial \bar{\mathbf{x}}} = \frac{1}{h_1} \bar{\mathbf{n}}_1 \ , \tag{2.29}$$

and analogously for the remaining nodes. For notational convenience, we summarize the shape functions into a vector $\mathbf{N} \in \mathbb{R}^3$ and write their partial derivatives as a matrix $\nabla \mathbf{N} \in \mathbb{R}^{3\times 2}$. As an appreciable property of the finite element approach pursued here[12], the shape functions and their derivatives are always evaluated with respect to the rest state such that they have to be computed only once. As explained in Sec. 2.1.3, the element coordinate system has to be aligned with the principal material axes before computing the derivatives. Since textiles are usually made of flat panels, an element's orientation with respect to the material axes is directly available and the alignment amounts to a single rotation in the material plane. For objects with curved rest states, this orientation cannot be determined automatically and has to be supplied by the user, e.g., via texture coordinates.

With the shape functions and their derivatives at hand, we can now proceed to the computation of discrete strains and stresses. We begin by introducing the interpolated geometry of the deformed element as

[12] In this so called *Total Lagrangian* formulation, the governing equations are expressed with respect to the rest state [Bat96].

$$\mathbf{x}(\bar{\mathbf{x}}) = \sum_{i=1}^{3} N_i(\bar{\mathbf{x}})\mathbf{x}_i \ , \qquad \text{where} \qquad \bar{\mathbf{x}} = (x, y) \in \bar{\Omega}_e. \tag{2.30}$$

With this expression, the discrete deformation gradient follows from Eq. (2.5) as

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \bar{\mathbf{x}}} = \sum_{i=1}^{3} \mathbf{x}_i \left(\frac{\partial N_i}{\partial \bar{\mathbf{x}}} \right)^t \ . \tag{2.31}$$

$\mathbf{F}$ is a (3×2)-matrix and, consequently, the Green strain $\mathbf{E} = \frac{1}{2}(\mathbf{F}^t\mathbf{F} - \mathbf{I})$ is a symmetric (2×2)-matrix. It should be stressed at this point that $\mathbf{E}$ refers to the local element space described in Sec. 2.1.3. This means that, regardless of the element's orientation in 3D space, the entries $\mathbf{E}_{ii}$ always refer to stretching along weft and warp directions while $\mathbf{E}_{ij}$ represents shear deformation. These properties also translate to the discrete elastic stress, which follows from Eq. (2.21). Viscous contributions are computed in a similar way upon replacing positions with velocities in Eq. (2.31) in order to obtain $\dot{\mathbf{F}}$. The strain rate $\dot{\mathbf{E}}$ is then computed from Eq. (2.24) and the discrete viscous stress follows from Eq. (2.25).

We now have everything in place to compute discrete nodal forces, which are given as[13]

$$\mathbf{f}^e = \int_{\Omega_e} \nabla\mathbf{N}\mathbf{S}\mathbf{F}^t \, da \ , \tag{2.32}$$

where $\mathbf{S}$ contains both viscous and elastic contributions and $\mathbf{f}^e \in \mathbb{R}^{3\times 3}$ denotes the matrix whose i-th row holds the internal force at node i. It can be seen from this expression that the deformation gradient accounts implicitly for the mapping between 2D element and 3D world space. Unlike in the co-rotational formulation , no additional transformations have to be computed, which translates into efficient force computations.

The derivatives of the linear shape functions are constant over the element, implying that $\mathbf{F}$ and $\mathbf{S}$ are constant as well. This simplifies the evaluation of the integral expression (2.32) and the nodal forces are obtained as

$$\mathbf{f}^e = \nabla\mathbf{N}\mathbf{S}\mathbf{F}^t h A_e \ , \tag{2.33}$$

where A_e is the area and h the thickness of the triangle in its rest state. This expression allows us to compute the discrete internal force due to a single element $\mathcal{K}_e$. The total force on a given node i is obtained by simply summing up contributions from all incident elements. Likewise, global Jacobians of elastic and viscous forces, which are required for implicit time integration, are assembled from element matrices. The latter are best derived component-wise by considering a single entry j of the force acting on node i,

$$\mathbf{f}^e_{ij} = \sum_{m=1}^{2} \mathbf{F}_{jm} \sum_{n=1}^{2} \mathbf{S}_{mn} \nabla\mathbf{N}_{in} \ . \tag{2.34}$$

[13] see Appendix A.2 for a derivation

The derivative with respect to component l of node k from the same element is

$$\frac{\partial \mathbf{f}_{ij}^e}{\partial \mathbf{x}_{kl}} = \sum_{m=1}^{2} \frac{\partial \mathbf{F}_{jm}}{\partial \mathbf{x}_{kl}} \sum_{n=1}^{2} \mathbf{S}_{mn} \nabla \mathbf{N}_{in} + \sum_{m=1}^{2} \mathbf{F}_{jm} \sum_{n=1}^{2} \frac{\partial \mathbf{S}_{mn}}{\partial \mathbf{x}_{kl}} \nabla \mathbf{N}_{in} \,. \tag{2.35}$$

For later reference, we denote the corresponding Jacobian matrix as $\mathbf{J}_{ik} = \mathbf{J}_{ik}^a + \mathbf{J}_{ik}^b$, accounting for the sum in the above expression. In order to compute $\mathbf{J}_{ik}$ for actual implementation, the remaining derivatives in (2.35) are readily determined analytically. However, an efficient implementation requires a careful analysis and arrangement of subterms in order to minimize arithmetic operations (see also [Mez08]). We leave such optimizations to the computer algebra software Maple[14], which automatically generates symbolically optimized code.

At this point, an important difference between the geometrically nonlinear approach pursued here and the co-rotational method should be made clear. In the latter, the force Jacobians of the elements are computed only once in their rest state and merely rotated to the current configuration. For concreteness, let $\bar{\mathbf{K}}_{ik} \in \mathbb{R}^{2\times 2}$ denote the submatrix of an element's rest state Jacobian that relates the force on node i to the displacement of node k. As described by Etzmuß et al. [EKS03], the matrix of the current configuration is computed as $\mathbf{K}_{ik} = \mathbf{R}\bar{\mathbf{K}}_{ik}\mathbf{R}^t$, where $\mathbf{R}^t \in \mathbb{R}^{2\times 3}$ is the transformation that rotates vectors in the element's current plane back to its rest state plane (xy-plane) and projects onto it (along the z-axis). This implies that any components along the current normal $\mathbf{n}$ of the element are filtered out by the transformation, i.e., $\mathbf{R}^t\mathbf{n} = \mathbf{0}$. In order to see the consequences of this, we will consider the simple example shown in Fig. 2.3.

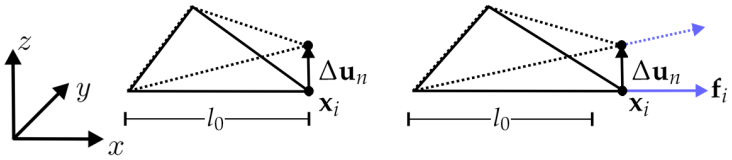


Figure 2.3: Effect of applying a small normal displacement to an element with zero (*left*) respectively nonzero (*right*) in-plane deformation.

We are interested in the relation between a displacement applied to node i and the resulting force $\mathbf{f}_i$ acting on it. In a state without in-plane deformation (see left of Fig. 2.3), a small displacement $\Delta\mathbf{u}_n = \epsilon\mathbf{n}$ normal to the element plane does – to first order – not lead to a change in force since it does not affect deformation[15]. The picture changes if the element is deformed in such a way that there is already a nonzero force $\mathbf{f}_i$ present. Applying the same displacement $\Delta\mathbf{u}_n$ in this state will again not change the magnitude of the force, but it *will* change its direction as shown on the right of Fig. 2.3. However, this change in force is not captured by the co-rotational formulation, since normal components of displacement are filtered out. For deformed elements, the co-rotational formulation thus erroneously predicts a zero change in force for out-of-plane displacement, $\mathbf{K}_{ik}\Delta\mathbf{u}_n = \mathbf{0}$, which can

[14] Maple is a computer algebra software by Waterloo Maple (see www.maplesoft.com).

[15] The deformation of a triangle element is a function of lengths of vectors and angles between vectors in its plane. Obviously, the gradients of these quantities lie in the element plane as well.

lead to stability problems[16]. This is not the case for the geometrically nonlinear approach, which can be seen by investigating the first term of (2.35). Considering the definition of the deformation gradient (2.31), we have

$$\frac{\partial \mathbf{F}_{jm}}{\partial \mathbf{x}_{kl}} = \delta_{jl} \nabla \mathbf{N}_{im} \,, \tag{2.36}$$

and, consequently, $\mathbf{J}^a_{ik}$ is a diagonal matrix. Provided that the current stress is such that the force on node i is nonzero, we therefore have $\mathbf{J}^a_{ik}\Delta\mathbf{u}_n \neq \mathbf{0}$. Likewise, it can be verified that $\mathbf{J}^b_{ik}\Delta\mathbf{u}_n = \mathbf{0}$ such that in total $\mathbf{J}_{ik}\Delta\mathbf{u}_n \neq \mathbf{0}$. The geometrically nonlinear formulation thus captures the effect of out-of-plane displacements, which translates into better stability for rotational motion.

It should be noted that the above described drawback of the co-rotational formulation is not as severe when simulating volumetric solids, since the nodal Jacobians of three-dimensional elements do not have non-trivial kernels in the rest state. For the case of deformable surfaces, however, the increased accuracy of the geometrically nonlinear approach is a strong reason to prefer it over the co-rotational variant.

2.2.2 Bending

The bending properties of textile materials have a significant impact on the formation of folds and thus overall visual appearance (see Fig. 2.4). A computational bending model has to be able to reproduce these effects in an accurate and computationally efficient way. Further requirements are good and intuitive control as well as consistent behavior across different discretizations and resolutions.

Thin-shell models are appealing since they possess sound mathematical and physical foundations. However, a finite element implementation of thin-shells requires a C^1-continuous displacement field, which necessitates either additional non-kinematic degrees of freedom (see e.g. [ZT00a, Bat96]) or elements with extended support [COS00, CO01]. Although nonlinear [GKS02] as well as linearized [TWS06] variants of the latter have been proposed in graphics, their increased complexity and computational costs have so far prevented a widespread use.

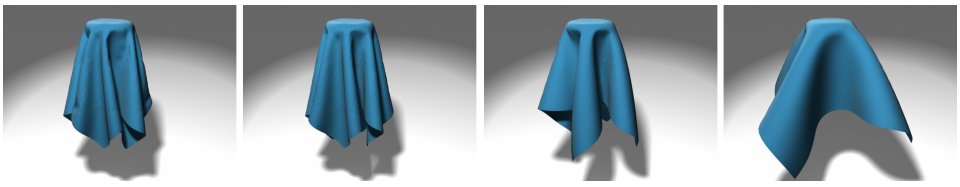


Figure 2.4: Drape tests with doubled bending stiffness from left to right.

Trading accuracy for efficiency, discrete bending models have traditionally been favored in graphics. The linear formulations described by Volino et al. [VMT06b] and Bergou et al. [BWH+06] promise high computational efficiency but offer only limited accuracy for large deformations. Similarly, the efficient cubic shell model by Garg et al. [GGWZ07] assumes isometric deformations, which

[16] see also the discussions in [CK02] and [VMTF09]

is reasonable for quasi-inextensible shells but too restrictive for general cloth animations. More general is the nonlinear model described in the contemporaneous works of Grinspun et al. [GHDS03] and Bridson et al. [BMF03], who compute bending energies and forces on hinge elements, consisting of two edge-adjacent triangles. In particular, the bending energy and thus the magnitude of the resulting forces depend only on the angle of the hinge, such that a clean separation of membrane and bending behavior is guaranteed. This is an appreciable property and we therefore adopt this model in our framework, using the expressions provided in [BMF03] in order to compute bending forces. For cloth materials, bending forces are typically weak and can therefore be integrated explicitly such that Jacobian matrices are not required.

2.2.3 Inertia and Dynamics

We have derived discrete internal forces on the basis of static equilibrium conditions (2.15, 2.16). The link to the dynamic setting is given by Newton's second law, which relates force to acceleration. Letting $\mathbf{f}_i$ denote the total force acting on a given node with mass m_i, we have

$$m_i \ddot{\mathbf{u}}_i = \mathbf{f}_i = \mathbf{f}_i^{ext} - \mathbf{f}_i^{el} - \mathbf{f}_i^{v} \ , \tag{2.37}$$

where $\mathbf{f}_i^{ext}$ denotes external forces such as gravity and $\mathbf{f}_i^{el}$ and $\mathbf{f}_i^{v}$ refer to internal elastic and viscous forces, respectively.

We compute nodal masses as $m_i = \rho A_i$, where ρ is the material's mass per unit area and A_i denotes the area associated with node i. The latter is determined from the areas of its adjacent faces A_{ij} as $A_i = \frac{1}{3}\sum_j A_{ij}$, noting that alternative tilings based on, e.g., Voronoi cells are possible as well [MDSB03a, EKS03].

Eq. (2.37) describes the nodal force balance for a single node. Defining the diagonal mass matrix $\mathbf{M}$ as

$$\mathrm{diag}(\mathbf{M})_{3(i+k)} = m_i \qquad \text{for } k \in \{1,2,3\} \text{ and } i \in \{1 \dots n_v\} \ ,$$

we can assemble the force balance equations for all nodes into a global system,

$$\mathbf{M}\ddot{\mathbf{u}}(t) = \mathbf{f}^{ext} - \mathbf{f}^{el}(\mathbf{u}(t)) - \mathbf{f}^{v}(\dot{\mathbf{u}}(t)) \ . \tag{2.38}$$

This formulation emphasizes the fact the elastic and viscous forces depend primarily on displacements and velocities, respectively. Expression (2.38) describes a set of $3n_v$ second order ordinary differential equations (ODEs) with respect to time. Together with starting positions and velocities, an initial value problem is obtained whose solution describes the continuous trajectories of the system's nodes. Discrete approximations of these trajectories are computed by means of numerical time integration, which is the subject of the next section.

2.3 Time Integration

The last section has lead to the spatially-discrete equations of motions that describe the dynamic evolution of the system as an initial value problem. The numerical solution of such problems is an intensively studied field and we generally refer to

the standard textbooks by Hairer et al. [HWN08, HW02, HLW06] and Ascher and Petzold [AP98] for detailed and comprehensive overviews. An analysis of time integration schemes for physically-based simulation in computer graphics can be found in the works by Hauth et al. [HES03, Hau04], Volino et al. [VMT01] and Boxerman [Box03].

Most integration schemes are only applicable to first order ODEs such that we have to transform Eq. (2.38). To this end we introduce nodal velocities $\mathbf{v} = \dot{\mathbf{u}}$ as additional variables and use $\mathbf{x} = \bar{\mathbf{x}} + \mathbf{u}$ for uniform notation. This allows us to express (2.38) as a coupled system of first order differential equations,

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{v} \\ \mathbf{M}\dot{\mathbf{v}} &= \mathbf{f}^{ext} - \mathbf{f}^{el}(\mathbf{x}) - \mathbf{f}^{v}(\mathbf{v}) \ . \end{aligned} \tag{2.39}$$

For brevity, we introduce the concatenated state vector $\mathbf{y} = (\mathbf{x}, \mathbf{v})$ and write

$$\dot{\mathbf{y}}(t) = \mathbf{f}(\mathbf{y}(t)) \ , \quad \text{where} \quad \mathbf{f} = \frac{d}{dt}\mathbf{y} \ . \tag{2.40}$$

This is a first order ODE and thus amenable to standard integration schemes. Its exact solution is

$$\mathbf{y}(t) = \mathbf{y}(t_0) + \int_{t_0}^{t} \dot{\mathbf{y}}(t) \ dt \ , \tag{2.41}$$

where t_0 corresponds to the initial state and $\mathbf{y}(t_0) = (\bar{\mathbf{x}}, 0)$ are initial values. Given a step size h, a numerical integration scheme computes approximations to the true solution at discrete instants in time $t_0 + n \cdot h$ as

$$\mathbf{y}^n \approx \mathbf{y}(t_0 + n \cdot h) \ . \tag{2.42}$$

We will briefly discuss some of the most widely used integration schemes in the following, focusing on methods with direct relevance to this work.

2.3.1 Explicit Methods and Stability

Explicit schemes compute the new state $\mathbf{y}^{n+1}$ of a system as a function of its current and possibly past states, requiring one or several evaluations of $\mathbf{f}$. The simplest representative of this class is the explicit Euler scheme, whose update formula is

$$\mathbf{x}^{n+1} = \mathbf{x}^n + h\mathbf{v}^n \tag{2.43}$$

$$\begin{aligned} \mathbf{v}^{n+1} &= \mathbf{v}^n + h\mathbf{f}(\mathbf{x}^n, \mathbf{v}^n) \\ &= \mathbf{v}^n + h\mathbf{M}^{-1}(\mathbf{f}^{ext} - \mathbf{f}^{el}(\mathbf{x}^n) - \mathbf{f}^{v}(\mathbf{v}^n)) \ . \end{aligned} \tag{2.44}$$

This method is only first order accurate[17], but higher-order integrators such as the fourth order Runge-Kutta scheme can be implemented with little additional effort.

Explicit methods are simple to implement and the costs per step are low, but a practical limitation is their conditional stability. In the broadest sense, an integration scheme is called stable with respect to a given problem if the solution remains always bounded. For certain difficult problems, stable integration with

[17] A method is accurate of order p if the local error $||\mathbf{y}^n - \mathbf{y}(t_0 + nh)||$ is of order $O(h^{p+1})$ as $h \to 0$, see [HWN08].

explicit schemes requires small step sizes, which in turn decreases computational efficiency. Equations for which this behavior is observed are referred to as being *stiff* and, unfortunately, this is typical of elasticity problems such as those arising in the simulation of deformable surfaces. The concept of stability can be quantified with the help of Dahlquist's test equation

$$\dot{y}(t) = \lambda y(t) \,, \qquad y(0) = y_0 \,, \qquad \lambda \in \mathbb{C} \,, \tag{2.45}$$

whose analytic solution is $y(t) = e^{\lambda t} y_0$. This equation can be considered a prototype of a stiff mechanical system since

$$e^{\lambda t} = e^{at} e^{ibt}, \quad \text{where} \quad \lambda = a + ib, \quad a, b \in \mathbb{R} \,, \tag{2.46}$$

from which it can be seen that the cases $a < 0$ and $a = 0$ describe conservative and dissipative oscillators, respectively [AP98]. Integration schemes are called unconditionally stable or A-stable if the solution stays bounded regardless of the step size when applied to the test equation. This is clearly not the case for the explicit Euler method, which yields the solution $y^{n+1} = (1 + h\lambda)^n y_0$ such that stability is only given if $|1 + h\lambda| \leq 1$. Likewise, no other Runge-Kutta-type explicit method is $A-$stable [NS74], but many implicit schemes are.

2.3.2 Implicit Methods and Dissipation

In contrast to explicit methods, implicit schemes also include the solution itself as a variable in the update formula. As the simplest representative, the implicit or backward Euler method is defined as

$$\begin{aligned} \mathbf{x}^{n+1} &= \mathbf{x}^n + h\mathbf{v}^{n+1} && (2.47) \\ \mathbf{v}^{n+1} &= \mathbf{v}^n + h\mathbf{f}(\mathbf{x}^{n+1}, \mathbf{v}^{n+1}) && (2.48) \\ &= \mathbf{v}^n + h\mathbf{M}^{-1}(\mathbf{f}^{ext} - \mathbf{f}^{el}(\mathbf{x}^{n+1}) - \mathbf{f}^{v}(\mathbf{v}^{n+1})) \,. \end{aligned}$$

Like its explicit counterpart, the implicit Euler scheme is only first order accurate, but it is unconditionally stable even for stiff systems. A step of backward Euler requires the solution of a nonlinear system of equations, which is in general significantly more expensive than computing a single step of an explicit scheme. On the other hand, the better stability of implicit schemes allows taking larger time steps such that for stiff problems, the additional costs per step typically pay off in terms of overall computation times.

The arising nonlinear system can be solved with the iterative Newton-Raphson scheme, which requires the solution of a linear system in each iteration. An alternative approach that enjoys widespread use in graphics is the *semi-implicit* formulation of backward Euler [BW98], which only requires the solution of a single linear system. This is achieved by linearizing the problem at the current state. Using a Taylor series expansion of the elastic forces and truncating after first order terms gives

$$\mathbf{f}^{el}(\mathbf{x}^{n+1}) \approx \mathbf{f}^{el}(\mathbf{x}^n) + \frac{\partial \mathbf{f}^{el}(\mathbf{x}^n)}{\partial \mathbf{x}} \cdot \Delta\mathbf{x} \,, \tag{2.49}$$

where $\Delta\mathbf{x} = (\mathbf{x}^{n+1} - \mathbf{x}^n)$. Applying an analogous expansion to the viscous forces and substituting

$$\Delta\mathbf{x} = h\Delta\mathbf{v} \ , \quad \text{where} \quad \Delta\mathbf{v} = \mathbf{v}^{n+1} - \mathbf{v}^n \ , \tag{2.50}$$

yields a linear system for the unknown velocities,

$$\left[\mathbf{M} + h^2 \frac{\partial \mathbf{f}^{el}(\mathbf{x}^n)}{\partial \mathbf{x}} + h \frac{\partial \mathbf{f}^{v}(\mathbf{v}^n)}{\partial \mathbf{v}}\right] \mathbf{v}^{n+1} = \mathbf{M}\mathbf{v}^n + h(\mathbf{f}^{ext} - \mathbf{f}^{el}(\mathbf{x}^n) - \mathbf{f}^{v}(\mathbf{v}^n)) \ . \tag{2.51}$$

Having solved for $\mathbf{v}^{n+1}$, new positions $\mathbf{x}^{n+1}$ are obtained from Eq. (2.47). When applied to cloth simulation, the semi-implicit Euler method can yield very fast computation times if large step sizes are used [BW98]. But unfortunately, this combination provides only low accuracy, which manifests as overly damped results that lack detail. In particular, fine folds and wrinkles are largely suppressed in this way, which partly defeats the purpose of using high-resolution meshes. This artificial damping, which is also known as numerical dissipation, is inherent to many implicit schemes, but particularly pronounced for low-order linearized variants. As shown by Boxerman [Box03], the effect of artificial damping increases with higher material stiffness and larger step sizes. By contrast, resorting to higher-order integrators alleviates dissipation [HE01a, CK02].

Higher-order implicit methods can be divided into two categories. *Multi-stage* schemes such as the family of implicit Runge-Kutta methods can be constructed up to arbitrary order, but require the solution of multiple systems of equations per time step [HW02]. By contrast, *multi-step* methods such as the Backward Differential Formulas (BDF) require only the solution of a single system per step. Since, these methods are based on an extrapolation of past states, they are only applicable to continuous functions. However, animations of deformable surfaces typically involve frequent collisions, which can constitute harsh discontinuities. Without modifications, multi-step methods cannot (or *should* not) be applied to such problems.

Another possibility to avoid numerical dissipation is by using variational integrators[18], which are renowned for their conservation properties [HLW06]. The latter is not to be confused with accuracy, which can be seen on the example of the *symplectic Euler* method,

$$\mathbf{v}^{n+1} = \mathbf{v}^n + h\mathbf{f}(\mathbf{x}^n, \mathbf{v}^n) \ , \tag{2.52}$$

$$\mathbf{x}^{n+1} = \mathbf{x}^n + h\mathbf{v}^{n+1} \ . \tag{2.53}$$

The first line is identical to the velocity update of the forward Euler method, but the positions are updated using the newly computed velocities. Although the symplectic Euler scheme is only first order accurate, it can be shown to preserve momentum and energy far better than the explicit Euler method [HLW06]. Similar arguments apply to implicit variational integrators such as the implicit midpoint rule, which is given as

$$\mathbf{v}^{n+1} = \mathbf{v}^n + h\mathbf{f}(\frac{\mathbf{x}^{n+1} + \mathbf{x}^n}{2}, \frac{\mathbf{v}^{n+1} + \mathbf{v}^n}{2}) \ , \tag{2.54}$$

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \frac{h}{2}(\mathbf{v}^{n+1} + \mathbf{v}^n) \ . \tag{2.55}$$

[18] More details on variational integrators are provided in Chapter 4.

While the computational costs per step are largely identical to the backward Euler method, the implicit midpoint scheme is second-order accurate and does not suffer from numerical dissipation. But despite these apparent advantages, the implicit Euler scheme is still preferred over the implicit midpoint rule – at least in computer animation – since especially semi-implicit implementations exhibit better stability.

Discussion To summarize, there is no single ideal approach for time stepping the dynamics of cloth and one has to balance accuracy against computation times when choosing an integration scheme. We pursue two strategies in this work.

The first one relies on the semi-implicit Euler method, but alleviates numerical damping by (*a*) using a material model that combines reduced elastic stiffness with deformation constraints and (*b*) using step sizes in the range of 10^{-3} seconds.

As the second approach, we address the problem directly and leverage asynchronous explicit integration in order to obtain cloth simulations without artificial damping.

Before proceeding to the next section, it should be emphasized that both choices fit well with the geometrically nonlinear membrane model described in the previous sections. Using a semi-implicit scheme requires only the solution of a single linear system and the computational costs do not differ much between our approach and the co-rotational variant. Furthermore, Kikuuwe et al. [KTY09] have reported good accuracy and efficiency for a similar approach in the context of three-dimensional elasticity. As for the explicit integration scheme, the costs of computing element forces are even slightly lower for the geometrically nonlinear approach since no rotations have to be computed. This observation was also made by Mezger [Mez08] for the case of volumetric solids.

2.4 Collision Handling

Beyond purely academic purposes, cloth animations generally involve collisions and contact. For example, cloth easily folds into configurations with multiple layers in close proximity and treating a single collision often induces multiple secondary collisions. Similarly, interactions between cloth and rapidly moving characters require special care in order to avoid intersections or other visual artifacts. Clearly, a robust method for handling collisions is vital for the visual quality and thus the success of cloth animations.

This section describes the approach used in this work, which is also the basis for the extensions toward asynchronous cloth simulation and parallel collision handling as described in Chapters 4 and 5, respectively. It relies on k-DOP[19] hierarchies [KHM+98, MKE03] for collision detection in combination with impulse-based collision response [Pro97, BFA02]. Both components are described in the following.

2.4.1 Collision Detection with k-DOP Hierarchies

A bounding volume hierarchy H is essentially a tree of bounding volumes. For a given triangle mesh, each of the nodes $N \in H$ is associated with a subset of the

[19] DOP stands for 'discrete oriented polytope' [KHM+98].

triangles and a bounding volume that encloses the corresponding geometry. In particular, the bounding volume of a given node encloses the bounding volumes of all of its children.

k-DOPs are a particular class of bounding volumes. A k-DOP D is a convex polyhedron, which is defined as the intersection of k half-spaces S_i

$$D = \cap_i S_i \qquad \text{, with } \; S_i = \{\mathbf{x} \in \mathbb{R}^3 | \mathbf{n}_i^t \mathbf{x} < d_i \in \mathbb{R}\} \; , \tag{2.56}$$

where $\mathbf{n}_i$ is the normal vector of the corresponding separating plane with distance d_i from the origin. The orientations of the planes are restricted to a discrete set of normal vectors, whose components can take on only the values $\{-1, 0, 1\}$. Additionally, the normals are chosen such that there are $k/2$ pairs of parallel planes. In three-dimensional space, the simplest representative is a 6-DOP, which corresponds to an axis-aligned bounding box (AABB) if the normals are chosen to coincide with the positive and negative directions of the three principal axes. Similar to AABBs, testing two $k-$DOPs for overlap amounts to checking the corresponding $k/2$ intervals, where disjointness can be signaled as soon as a non-overlapping interval is found. Since bounding volumes only approximate their enclosed geometry, an overlap between two DOPs does not necessarily imply that there are actual intersections. However, the probability of encountering such *false positives* can be reduced by using larger values of k, which leads to more tight-fitting bounding volumes that can better adapt to arbitrarily oriented geometry. We use $18-$DOPs in our implementation and construct corresponding hierarchies with a top-down approach as described by Mezger et al. [MKE03]: starting with a single DOP that encloses the entire geometry of a given (rigid or deformable) object, the bounding volumes are recursively split along their longest axis until all DOPs contain only a single triangle.

Testing two meshes with hierarchies H_A and H_B for interference is a recursive process. The first test determines whether the bounding volumes of the two root nodes overlap and if this is the case, further pair-wise tests are applied recursively among their children. The result of this recursion is a (possibly empty) set of triangle pairs, corresponding to leaf nodes with overlapping bounding volumes. Note that these pairs constitute only *potential* collisions such that further testing is required in order to determine truly intersecting triangles, see Sec. 2.4.2.

The procedure outlined above works well if the two hierarchies H_A and H_B belong to different meshes, as is the case when detecting collisions between cloth and rigid objects. When used for detecting self-collisions, however, this approach is less efficient since the bounding volumes of adjacent regions necessarily overlap, even for completely flat configurations. We employ normal cones [Pro95] in order to skip test for such flat regions and additionally use adjacency-based culling to eliminate unnecessary distance computations for neighboring triangles. These modifications greatly accelerate computations and are completely transparent to the parallel algorithm described in Chapter 5.

The time integration scheme produces geometry output at discrete points in time. For collision detection, however, taking into account only a single configuration is not sufficiently robust since interferences that occur in between two states (i.e., during a time step) can pass unnoticed. Such interferences can be captured when enlarging the bounding volumes such that they enclose both the geometry

at the beginning and at the end of a given time step. Applying a recursive test as usual to these swept hierarchies yields a set of triangle pairs, whose trajectories have to be checked for true intersection. This process is commonly referred to as *continuous collision detection* whereas the term *proximity detection* is used when only a single configuration is taken into account.

2.4.2 Impulse-based Collision Response

A collision detection pass with bounding volume hierarchies yields a set of potentially colliding triangle pairs, which require further processing in order to determine actual intersections. Different treatments are necessary depending on whether a pair originates from proximity or continuous collision detection. Both cases are described in the following paragraphs.

As the input for collision response, we assume that position vectors $\mathbf{x}$ and $\tilde{\mathbf{x}}$ are supplied, corresponding to configurations at the beginning and end of a given time step, respectively. Furthermore, it is assumed that the velocities $\mathbf{v}$ are such that $\tilde{\mathbf{x}} = \mathbf{x} + h\mathbf{v}$, where h is the step size.

Proximity-based Response For each pair of triangles in close proximity, we first perform a geometric test [Möl97] in order to determine the distance between their closest points. If this value is larger than a given proximity threshold[20] ε_c, the triangles are considered as non-colliding. Otherwise, exhaustive tests are performed on a set of primitive pairs, which consists of the six vertex-triangle and nine edge-edge combinations of the two triangles.

To simplify notation, we let $\mathbf{x}_i$ with $i = 1 \dots 4$ denote the positions of the four vertices of a given primitive pair at the beginning of the time step. For conciseness we will only give expressions for the vertex-triangle case, but refer to the generic pair of closest points as $\mathbf{x}_a$ and $\mathbf{x}_b$ in order to allow easy translation to edge-edge collisions. For a vertex-triangle collision, the indices $i = 1 \dots 3$ refer to the triangle's vertices whereas $\mathbf{x}_4$ denotes the single vertex. The latter is also one of the two closest points and we set $\mathbf{x}_b = \mathbf{x}_4$. The other closest point $\mathbf{x}_a$ is computed as a convex combination of the triangle's vertices using a geometric point-triangle test [AMHH08], which results in a set of barycentric weights λ_i such that

$$\mathbf{x}_a = \sum_{i=1}^{3} \lambda_i \mathbf{x}_i \ , \qquad \text{where} \ \sum_{i=1}^{3} \lambda_i = 1 \ . \tag{2.57}$$

If the distance $d_{ab} = ||\mathbf{x}_b - \mathbf{x}_a||$ between the closest points is smaller than the threshold value ε_c, further approach between the primitives has to be prevented. We use stopping impulses for this purpose that are computed in the following way. Let $\mathbf{v}_a$ and $\mathbf{v}_b$ denote velocities corresponding to the closest points, where the former is computed in analogy to Eq. (2.57). The relative velocity in normal direction is determined as $\hat{v}_r = \mathbf{n}^t(\mathbf{v}_b - \mathbf{v}_a)$, where the sign of the triangle's normal $\mathbf{n}$ is chosen such that $\hat{v}_r > 0$ if the two points are approaching. If $\hat{v}_r < 0$, the primitives are separating and no action is required. Otherwise, a pair of normal impulses $\hat{\mathbf{p}}_b = \hat{p}_4\mathbf{n}$ and $\hat{\mathbf{p}}_a = -\hat{\mathbf{p}}_b$ is computed such that, when applied to $\mathbf{v}_b$ and

[20] The constant ε_c effectively models the thickness of the cloth surface.

$\mathbf{v}_a$, they cancel the relative normal velocity. Letting $\hat{v}_i = \mathbf{n}^t\mathbf{v}_i$ and $\hat{p}_i$ denote normal velocities and corresponding impulses of the four vertices, these conditions can be expressed as

$$(\hat{v}_4 + \frac{\hat{p}_4}{m_4}) - \sum_{i=1}^{3} \lambda_i(\hat{v}_i + \frac{\hat{p}_i}{m_i}) = 0 \qquad \text{subject to} \qquad \sum_{i=1}^{3} \hat{p}_i = -\hat{p}_4 \; . \tag{2.58}$$

In order to solve for the unknown impulses, additional assumptions have to be made on the way in which they are distributed to the vertices of the triangle. A simple way of distribution is to use the barycentric weights as suggested in [BFA02], i.e., to define $\hat{p}_i = -\lambda_i\hat{p}_4$, which ensures continuity as a collision moves from a given triangle to a neighboring one. This leaves the normal impulse $\hat{p}_4$ as the only unknown, which is determined from Eq. (2.58) as

$$\hat{p}_4 = -\hat{v}_r \left(\sum_{i=1}^{3} \frac{\lambda_i^2}{m_i} + \frac{1}{m_4} \right)^{-1} \; . \tag{2.59}$$

The stopping impulses filter the velocities of the vertices such that a further approach is prevented. In order to maintain the separating distance ε_c, repelling impulses are added that push primitives in too close proximity apart. These are computed in analogy to Eq. (2.59), essentially creating an appropriate amount of negative relative velocity. Finally, Coulomb-like friction[21] is incorporated by modifying the tangential velocities $\bar{\mathbf{v}}_i = \mathbf{v}_i - \mathbf{n}^t\mathbf{v}_i$ in proportionality to the applied normal impulses.

Continuous Collision Response Pairs of triangles that originate from a pass of continuous collision detection can potentially interfere at any time during the interval $[0, h]$. In analogy to the case of proximities, the detection and treatment of actual continuous collisions is again based on a decomposition into primitive pairs. A necessary condition for a pair of primitives to collide during the interval is that its four vertices are coplanar for some $t \in [0, h]$. This can be expressed in a formal way using the observation that the four vertices of a given primitive pair describe a tetrahedron. Coplanarity is then equivalent to a vanishing volume of the tetrahedron, which is computed as

$$V(t) = \frac{1}{6}[\mathbf{e}_{21}(t) \times \mathbf{e}_{31}(t)]^t\mathbf{e}_{41}(t) \; , \tag{2.60}$$

where $\mathbf{e}_{ij}(t) = (\mathbf{x}_i + t\mathbf{v}_i) - (\mathbf{x}_j + t\mathbf{v}_j)$. Requiring $V(t) = 0$ yields a cubic equation in t, which is solved for possible times of coplanarity t_i. The valid roots $t_i \in [0, h]$ are then processed in ascending order and proximity tests are performed on the geometry at the corresponding times. Primitive collisions detected in this way are then treated with stopping impulses as explained in the previous paragraph.

[21] Coulomb friction is an empirically derived model which distinguishes between static (resting) and kinetic (sliding) friction, for which independence of the sliding velocity is postulated, see also [Pop10]. Letting $\mu_s$and μ_k denote coefficients of friction and $\mathbf{f}_n$ the normal force acting between the surfaces, the tangential contact forces for the static and kinetic case are defined as $\mathbf{f}_s \leq \mu_s\mathbf{f}_n$ and $\mathbf{f}_k = \mu_k\mathbf{f}_n$, respectively.

2.4.3 Iterative Collision Resolution

As described by Bridson et al. [BFA02], proximity-based and continuous collision handling can be combined into a robust framework for iterative collision resolution. The original method consists of three stages, but we use only two of them in our implementation. In the first stage, collisions are detected and handled based on proximities at the beginning of a time step. In the second stage, continuous collision detection and response passes are applied iteratively until all remaining collisions are resolved. An additional third stage with rigid impact zones [Pro97] or the method by Harmon et al. [HVTG08] can be used in order to treat cases when the second stage is unable to resolve all collisions. While such a fail-safe can give increased robustness for large time steps or high coefficients of friction, we found the first two stages to be sufficient for all examples considered in this work.

This iterative collision resolution framework is a state-of-the-art approach and compelling results can be obtained even for challenging animations with massive and intricate self-collisions. Its robustness with respect to such extreme conditions comes, however, at the price of high computational costs, which can easily constitute the largest part of the simulation time. By way of illustration, we will consider a collision-intensive example, which is inspired by the collision handling stress test described in [HVTG08].

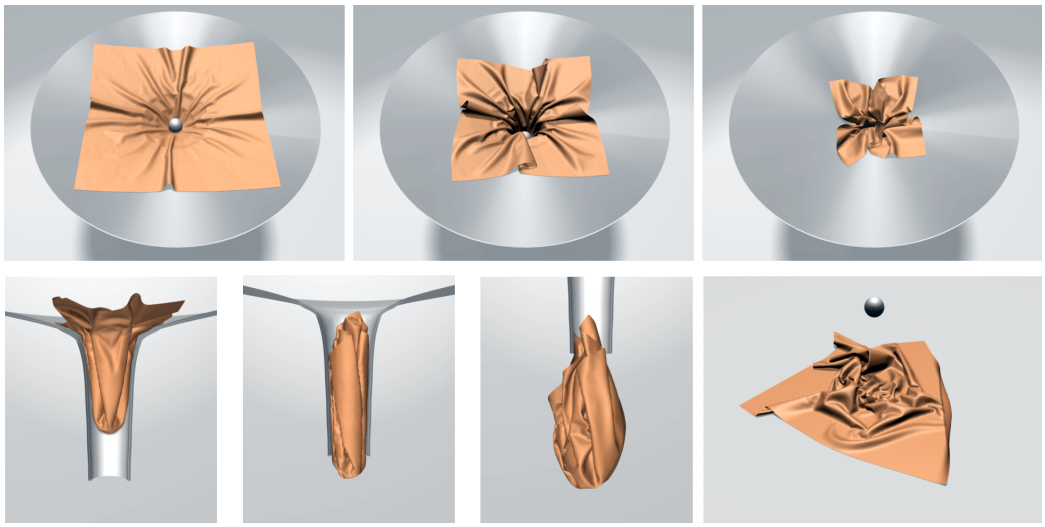


Figure 2.5: A *stress test* for collision handling: a piece of cloth is forced through a narrow funnel.

A square cloth with 14 476 faces is forced to follow the motion of a rigid sphere that passes through a narrow funnel. Some representative frames of the four seconds of animation are shown in Fig. 2.5. The cloth first settles on the upper part of the funnel, but is soon pulled inwards due to the scripted motion of the sphere. Complex folds start to form and merge into tightly packed layers as the sphere moves futher down, inducing massive self-collisions. The cloth starts to unfold when the sphere leaves the funnel and finally settles on the rigid floor. While the first part of the animation requires few intervention from the second stage, up to ten iterations are required in order to resolve all collisions while the cloth

passes through the funnel. This translates into relatively high computational costs: using a step size of $0.001s$ a single 25Hz frame took 54.72 seconds to compute[22] on average and 72 percent of this time were spent on collision handling. While this example is more collision-intensive than the average case, it is not unusual that collision handling takes up half of the simulation time and performance improvements are thus of immediate practical relevance. This motivates our work on parallel collision handling, which is presented in Chapter 5.

2.5 Summary

This chapter has laid out a computational framework for accurate physical cloth simulation. Being one of its central components, we have advocated a nonlinear continuum mechanics approach for the in-plane behavior in order to accomodate large rotational deformations and direction-dependent material properties. As for the latter, we restricted our considerations to a simple anisotropic material law, anticipating the extensions to be introduced in Chapter 3. Discrete in-plane forces were derived using linear triangle finite elements and, as is common practice in cloth simulation, we decided on a discrete hinge-based bending model. In order to solve the spatially-discrete equations of motion in time, we have subsequently discussed several numerical integration schemes. Despite certain reservations with respect to accuracy, we selected the semi-implicit Euler scheme as the default method, postponing an alternative approach to Chapter 4. Finally, we have addressed collision handling and described the sequential basis for the parallel approach presented in Chapter 5.

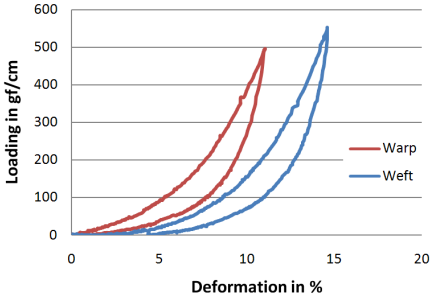
[22] Times were measured on a laptop computer with an Intel Core2Duo processor running at 2.0GHz. Only a single core was used.

Chapter 3

Deformation Constraints for Biphasic Anisotropic Cloth

Many cloth materials show a direction-dependent and highly nonlinear response to in-plane deformations. Although this behavior is not well approximated by linear material laws, most animation methods avoid the complexity of fully nonlinear models for the sake of computational efficiency. This chapter presents an alternative approach that combines a simple elastic material law and geometric deformation constraints into an efficient model for *biphasic* and *anisotropic* cloth.

3.1 Introduction



Textiles and especially woven fabrics are complex materials that exhibit anisotropic, highly nonlinear, and even hysteretic[1] properties as shown in the example to the right. Much of this complexity is due to the material behavior and surface structure at the yarn level, but also the weave pattern plays a role in this context. As a common characteristic of most textiles, the resistance to stretching is initially weak but a high stiffness is observed for larger strains. Consequently, cloth is readily stretched by a few percent but deformations beyond material-specific thresholds do not occur under normal circumstances. Due to the anisotropic nature of cloth, these effective limits can vary significantly for deformations in different directions.

A widespread approach in computer graphics is to model the elastic response of textiles with linear material laws [BW98, CK02]. Since this model offers only a single stiffness parameter for the entire deformation range, a high value is required in order to comply with the material's effective strain limits. This, however, is a poor approximation of general cloth, especially for the small deformation range, and a lot of its characteristic behavior is lost in this way. Furthermore, highly stiff

[1] In this context, *hysteresis* denotes the effect that measured force-deformation curves do not coincide for loading and unloading directions.

materials degrade the efficiency of the physics solver [HE01a] and amplify the effects of numerical dissipation [Box03]. A second possibility is to use nonlinear material laws, which can be described, e.g., in form of stress-strain curves based on measured data. While this approach is accurate, its high nonlinearity calls for more elaborate numerical solvers in order to allow stable time integration. This, in turn, leads to an increase in complexity as well as computation times.

An attractive alternative in this context is to replace the stiff component of the nonlinear material law by a geometric *constraint* and to use a soft elastic material for the non-stiff part. The resulting *biphasic* model captures the characteristic stretching properties of cloth much better than a linear material. At the same time, numerical complications arising from stiff and nonlinear models are avoided.

Overview and Contributions Drawing on this concept, the present chapter introduces *Continuum-based Strain Limiting* (CSL), which is a novel approach for the simulation of anisotropic biphasic cloth. We impose deformation constraints on a continuum-based strain measure, which allows us to accurately distinguish between all deformation modes. In particular, individual limits can be used for stretching in weft and warp directions as well as shearing, thus enabling full control over deformations. Constraints are formulated per triangle on the basis of the co-rotated Cauchy strain tensor, which is discretized with linear finite elements. Unlike previous approaches, our method is thus able to enforce anisotropic deformation constraints on general unstructured triangle meshes.

The remainder of this chapter describes all components of this approach in detail. After a short review of existing work on deformation limiting, we turn to the central part of this chapter. We first describe how to formulate and enforce deformation constraints on triangle elements (Sec. 3.3.1), subsequently address the combination of elemental responses into a global solution (Sec. 3.3.2), and finally discuss the question of how to appropriately choose material coefficients and deformation limits (Sec. 3.3.3). We demonstrate the qualitative capabilities of continuum-based strain limiting in Sec. 3.4 and compare its performance to conventional approaches. The chapter concludes with a summary and a discussion of limitations as well as directions for future work.

3.2 Previous Work

The idea to combine elastic forces with deformation constraints has been pursued by several previous works. The first approach in this direction goes back to Provot [Pro95], who described a deformation limiting method for mass-spring systems in the context of explicit time integration. Using weaker springs allows the use of larger step sizes but also entails unrealistically high stretch deformations. In order to reduce this unwanted effect, Provot suggested to correct end points of overly strained edges in an iterative manner. A deformation threshold of 10% was established and subsequent work largely followed this example. Bridson et al. [BFA02] extended Provot's technique to strain-rate limiting, recasting the method into a velocity-correcting formulation. In order to reduce bias resulting form a fixed edge ordering, a randomized traversal was suggested. Similar in spirit is the

work by Müller et al. [MHHR06, Mül08], who used general position constraints as the basic simulation principle.

In contrast to these iterative methods, Hong et al. [HCJ+05] couple edge length constraints globally using a Lagrangian mechanics formulation and require constraint satisfaction at the end of each time step. Similar to the semi-implicit Euler scheme of Baraff and Witkin, they linearize the constraint maintaining forces and solve a single linear system. However, constraints can only be satisfied to first order in this way, which can lead to large violations especially for rotational motion.

Tsiknis [Tsi06] describes a three step scheme consisting of constraint force estimation, strain limiting and global response. The second step forms an exception to conventional strain limiting, as it works on triangles instead of edges. However, only one principal strain direction is considered and anisotropic properties cannot be taken into account. The first and the third step rely again on edge-based techniques such that the associated limitations are inherited.

While the above approaches settled for a 10% deformation limit, an efficient way to simulate quasi-inextensible cloth was introduced with the fast projection method by Goldenthal et al. [GHF+07]. Since constraint satisfaction is guaranteed at the end of each time step, very strict deformation limits can be enforced with this technique. Based on fast projection, English et al. [EB08] animate inextensible triangle meshes, enforcing zero in-plane deformation via edge length constraints. Note that for such quasi-inextensible materials, anisotropy is not of primary importance since deformations are assumed to be very small. For many general textile materials, however, this assumption does not hold.

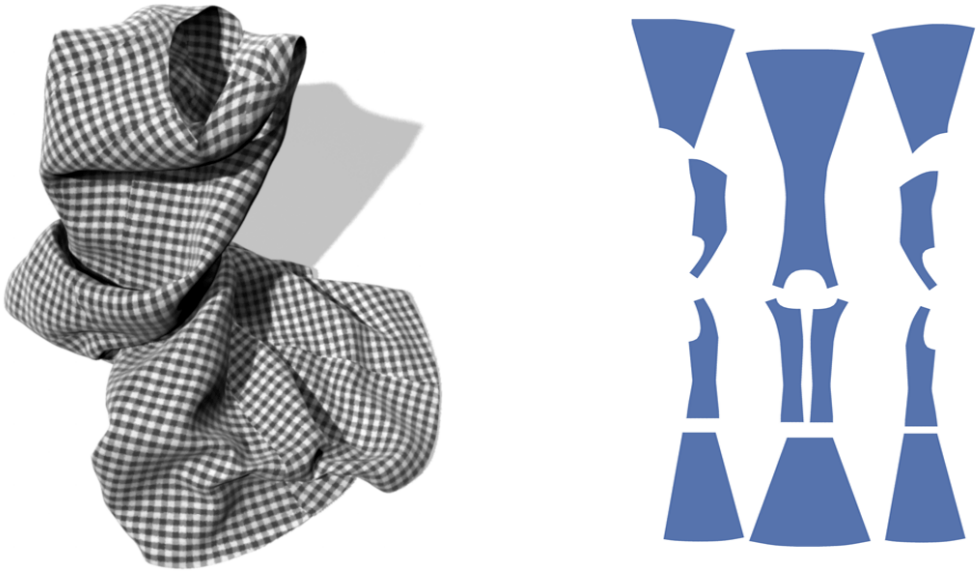


Figure 3.1: A complex garment (*left*) made of 13 flat panels with curved boundaries (*right*) is modeled and animated with an unstructured triangle mesh. The orientation of the panels corresponds to their alignment with the material's weft and warp directions, running along the horizontal and vertical axes of the image plane, respectively.

A common point of all previous approaches is that constraints are formulated and/or enforced on edges, making them more or less strongly dependent on discretization. Since an edge can only respond to deformation along its direction, a regular weft-warp-aligned mesh is necessary to model anisotropic deformation limits. This is a significant disadvantage for practical animations of garments, which are typically made of complex panels with curved boundaries (see Fig. 3.1).

The method presented in this chapter departs significantly from previous approaches. In order to resolve the above shortcomings, constraints are formulated on the basis of a continuous deformation measure. Consequently, CSL allows accurate control of stretch and shear deformations with anisotropic limits, regardless of the underlying mesh.

3.3 Continuum-based Strain Limiting

This section describes the formulation and enforcement of continuum-based deformation constraints.

Instead of directly coupling elastic forces and constraints, we implement the biphasic model as a two-step process: in the first step, the system is advanced in time considering only physical forces, not constraints. The resulting positions and velocities are the input to the second step, which enforces deformation constraints by modifying velocities appropriately. This approach facilitates time integration and is in line with previous work [Pro95, BFA02, GHF+07].

The physical forces of the first step include the elastic part of the biphasic model, for which we use the approach described in Chapter 2. The second step consists again of two building blocks: one is the *local* constraint enforcement on triangle elements, the other is the combination of element responses into a *global* solution. We pursue an iterative global enforcement scheme, which corrects individual elements in isolation until all deformation constraints are satisfied. We will start the detailed description with the local part of this approach, i.e., the formulation and enforcement of deformation constraints on triangle elements.

3.3.1 Deformation Constraints on Triangle Elements

In the course of a global deformation limiting step, elemental constraints have to be evaluated and enforced multiple times for each triangle. Optimizing this kernel will pay off in terms of overall performance and the choice of the deformation measure is a central component in this context. A linear measure promises good computational efficiency and we therefore decide to use the Cauchy strain as the basis for the deformation constraints. However, since the triangles of a cloth mesh will generally undergo large rotations we use the co-rotational formulation. This amounts to extracting rotations from the displacements before computing the strain, which is explained in the following paragraph.

Extracting Rotations Consider a triangle element with rest state nodal positions $\bar{\mathbf{x}}_i \in \mathbb{R}^2$ which have been translated, rotated, and deformed to current positions $\mathbf{x}_i \in \mathbb{R}^3$ as illustrated in Fig. 3.2. We want to extract the rotational part from this transformation and will do so on the basis of the corresponding deformation gradient $\mathbf{F}$ or, more precisely, its polar decomposition [GVL96].

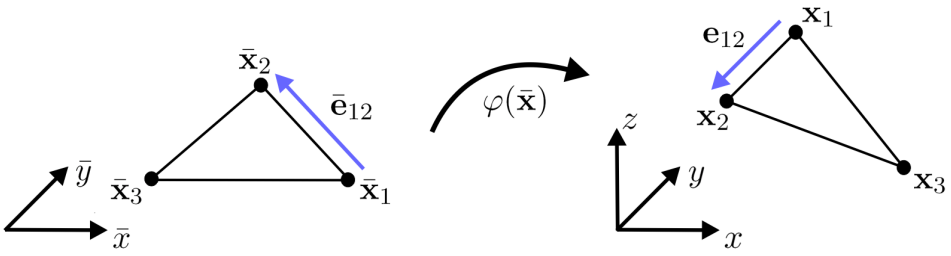


Figure 3.2: A triangle is transformed from its two-dimensional rest state (*left*) to its current configuration in three-dimensional space (*right*). The transformation φ generally includes translation and rotational components, as well as pure deformations.

Assuming deformations to be piecewise constant, we compute $\mathbf{F}$ for a given triangle using the linear finite element discretization as described in Sec. 2.2.1,

$$\mathbf{F} = \frac{\partial \varphi(\bar{\mathbf{x}})}{\partial \bar{\mathbf{x}}} = \sum_{i=1}^{3} \mathbf{x}_i \left(\frac{\partial N_i}{\partial \bar{\mathbf{x}}} \right)^t .$$

Recall that $\mathbf{F} \in \mathbb{R}^{3\times 2}$ is a linear operator that transforms vectors from the two-dimensional rest space of an element to their deformed counterparts in three-dimensional space. In particular, we can express the deformed edges $\mathbf{e}_{ij} = \mathbf{x}_j - \mathbf{x}_i$ of the triangle as

$$\mathbf{e}_{ij} = \mathbf{F}\bar{\mathbf{e}}_{ij} , \qquad \text{where} \quad \bar{\mathbf{e}}_{ij} = \bar{\mathbf{x}}_j - \bar{\mathbf{x}}_i . \tag{3.1}$$

With this relation, the polar factorization of the deformation gradient, $\mathbf{F} = \mathbf{RU}$, can be interpreted in the following way: a given edge $\bar{\mathbf{e}}_{ij}$ is first stretched and sheared in the plane of the element according to the deformation matrix $\mathbf{U} \in \mathbb{R}^{2\times 2}$, leading to an intermediate vector $\tilde{\mathbf{e}}_{ij} = \mathbf{U}\bar{\mathbf{e}}_{ij}$. This vector is then rotated to its current orientation in three-dimensional space as $\mathbf{e}_{ij} = \mathbf{R}\tilde{\mathbf{e}}_{ij}$, where $\mathbf{R} \in \mathbb{R}^{3\times 2}$ is the element's effective rotation matrix. We follow Etzmuß et al. [EKS03] in order to determine this rotation, i.e., we first compute $\mathbf{U}$ from $\mathbf{F}^t\mathbf{F} = \mathbf{U}^2$ using singular value decomposition and then obtain $\mathbf{R} = \mathbf{FU}^{-1}$. Since $\mathbf{U}$ is only a (2×2)-matrix, its singular values and its inverse can be computed very efficiently. Having determined $\mathbf{R}$, we use its transpose to rotate the element's positions back to their rest state plane, by construction corresponding to $z = 0$, and transform to two dimensions by discarding the third components. We thus obtain rotation-free in-plane displacements,

$$\mathbf{u}_i = \mathbf{R}^t\mathbf{x}_i - \bar{\mathbf{x}}_i , \qquad \text{where} \quad \mathbf{u}_i \in \mathbb{R}^2 ,$$

from which we finally compute the linear Cauchy strain of the element as

$$\boldsymbol{\varepsilon} = \sum_{i=1}^{3} \mathbf{B}_i \mathbf{u}_i = \mathbf{B}\mathbf{u}^e \,. \tag{3.2}$$

Here and henceforth, the superscript e denotes a six component vector holding nodal quantities such as displacements, $\mathbf{u}^e = [\mathbf{u}_1 \ \mathbf{u}_2 \ \mathbf{u}_3]^t$, whereas $\mathbf{B} \in \mathbb{R}^{3\times 6}$ denotes the concatenation of the three strain-displacement matrices

$$\mathbf{B}_i = \begin{bmatrix} \frac{\partial N_i}{\partial x} & 0 \\ 0 & \frac{\partial N_i}{\partial y} \\ \frac{1}{2}\frac{\partial N_i}{\partial y} & \frac{1}{2}\frac{\partial N_i}{\partial x} \end{bmatrix} \,. \tag{3.3}$$

Having defined an appropriate deformation measure, we can now proceed to the actual constraints.

Formulating and Enforcing Constraints Suppose that the current deformation $\varepsilon(\mathbf{u}^e)$ of a given element, computed via Eq. (3.2), violates one or several of the strain limits, which are supplied in a vector $\varepsilon^{lim} \in \mathbb{R}^3$. In general, not all three limits are violated at the same time and it seems to be the least intervention to leave non-violated modes unchanged. To this end, we define a target deformation ε^{tar} for the element as

$$\varepsilon_i^{tar} = \begin{cases} \varepsilon_i^{lim} & \text{, if } \ \varepsilon_i > \varepsilon_i^{lim} \\ \varepsilon_i & \text{, otherwise.} \end{cases} \tag{3.4}$$

Limits for maximum compressive deformation are set in an analogous way. We now seek to compute correcting displacements $\Delta\mathbf{u}^e$ such that all deformation limits are satisfied at the same time, i.e.,

$$\varepsilon(\mathbf{u}^e + \Delta\mathbf{u}^e) = \varepsilon^{tar} \,. \tag{3.5}$$

Due to the linearity of ε we obtain a linear system for the unknown correcting displacements

$$\mathbf{B}\Delta\mathbf{u}^e = \varepsilon(\mathbf{u}^e) - \varepsilon^{tar} \,, \tag{3.6}$$

which describes three equations for six unknowns. In order to prevent momentum drift, we add three equations that explicitly require the corrections to be free from unwanted translational and rotational components. To this end, we start by expressing $\Delta\mathbf{u}^e$ in terms of equivalent correcting velocities $\Delta\mathbf{v}^e$. The latter are computed in such a way that, when applied retrospectively to the average velocities of a given time step, they yield the desired displacement corrections $\Delta\mathbf{u}_i = h\Delta\mathbf{v}_i$, where h is the step size of the time integration scheme. Requiring $\Delta\mathbf{v}^e$ to not affect linear momentum gives two additional equations,

$$\sum_{i=1}^{3} m_i \Delta\mathbf{v}_{i,x} = 0 \qquad \text{and} \qquad \sum_{i=1}^{3} m_i \Delta\mathbf{v}_{i,y} = 0 \,, \tag{3.7}$$

where m_i denotes the mass of node i. Similarly, a sixth equation is obtained by enforcing $\Delta\mathbf{v}^e$ to not change rotational momentum[2]. To this end, we define a set

[2]Since the corrections $\Delta\mathbf{v}^e$ are confined to the plane of the element, they can only cause rotational motion about its normal direction.

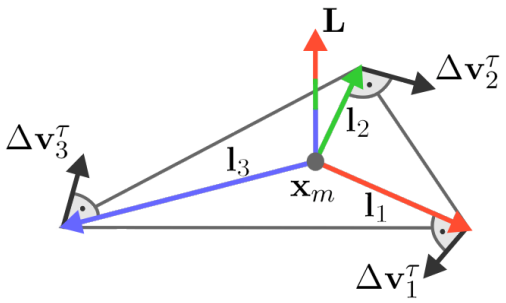


Figure 3.3: Rotational components $\Delta\mathbf{v}_i^\tau$ of correcting velocities and resulting angular momentum $\mathbf{L}$. The coloring indicates the contributions of individual nodes.

of vectors $\mathbf{l}_i = \mathbf{x}_i - \mathbf{x}_m$ that link the triangle's nodes with its center of mass $\mathbf{x}_m$ as shown in Fig. 3.3 and require

$$\sum_{i=1}^{3} m_i(\mathbf{l}_{i,x}\Delta\mathbf{v}_{i,y} - \mathbf{l}_{i,y}\Delta\mathbf{v}_{i,x}) = 0\,. \tag{3.8}$$

We thus have six equations for six unknowns and can now set up a linear system for elemental deformation limiting. Defining the right hand side vector $\mathbf{b} \in \mathbb{R}^6$ component-wise as

$$\mathbf{b}_i = \begin{cases} \varepsilon_i(\mathbf{u}^e) - \varepsilon_i^{tar} & \text{, for } i = 1\ldots3 \\ 0 & \text{, for } i = 4\ldots6 \end{cases} \tag{3.9}$$

we write the linear system of equations as

$$\mathbf{A}\Delta\mathbf{v}^e = \mathbf{b}\,, \tag{3.10}$$

where the matrix $\mathbf{A}$ is defined as

$$\mathbf{A} = \begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial x} & 0 & \frac{\partial N_3}{\partial x} & 0 \\ 0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial y} & 0 & \frac{\partial N_3}{\partial y} \\ \frac{1}{2}\frac{\partial N_1}{\partial y} & \frac{1}{2}\frac{\partial N_1}{\partial x} & \frac{1}{2}\frac{\partial N_2}{\partial y} & \frac{1}{2}\frac{\partial N_2}{\partial x} & \frac{1}{2}\frac{\partial N_3}{\partial y} & \frac{1}{2}\frac{\partial N_3}{\partial x} \\ m_1 & 0 & m_2 & 0 & m_3 & 0 \\ 0 & m_1 & 0 & m_2 & 0 & m_3 \\ -m_1\mathbf{l}_{1,y} & m_1\mathbf{l}_{1,x} & -m_2\mathbf{l}_{2,y} & m_2\mathbf{l}_{2,x} & -m_3\mathbf{l}_{3,y} & m_3\mathbf{l}_{3,x} \end{bmatrix}\,. \tag{3.11}$$

The matrix $\mathbf{A}$ is invertible for all non-degenerate triangles[3] such that this system has exactly one solution.

This solution of (3.10) has to be computed once for every over-deformed element in each iteration of the global enforcement scheme, which can become quite expensive if a conventional solver, for example based on Gaussian elimination, is used. The computational costs can, however, be significantly reduced using the

[3] In this case, non-degeneracy is equivalent to non-zero area.

observation that, except for the last row, all entries of $\mathbf{A}$ are rest state quantities. We first decompose the linear system and write

$$\begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_3 & \mathbf{A}_4 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix} , \tag{3.12}$$

where $\mathbf{A}_1$ is the upper-left (5×5)-submatrix and $[\mathbf{s}_1\ \mathbf{s}_2]^t$ and $[\mathbf{r}_1\ \mathbf{r}_2]^t$ are partitioned solution and right hand side vectors, respectively. Letting $\mathbf{S}$ denote the Schur complement [GVL96] of the submatrix $\mathbf{A}_1$,

$$\mathbf{S} = \mathbf{A}_4 - \mathbf{A}_3\mathbf{A}_1^{-1}\mathbf{A}_2 , \tag{3.13}$$

the solution of (3.12) can be recast into two steps:

$$\begin{aligned} \mathbf{s}_2 &= \mathbf{S}^{-1}(\mathbf{r}_2 - \mathbf{A}_3\mathbf{A}_1^{-1}\mathbf{r}_1) , \\ \mathbf{s}_1 &= \mathbf{A}_1^{-1}(\mathbf{r}_1 - \mathbf{A}_3\mathbf{s}_2) . \end{aligned}$$

Note that $\mathbf{S}$ is a scalar such that its inverse is obtained directly. Computing the (5×5)-inverse matrix $\mathbf{A}_1^{-1}$ is more involved, but since it only depends on rest state quantities it can be precomputed. Using an optimized code, only 1 division, 33 additions and 37 multiplications have to be performed at run time to solve (3.12). Compared to the solution via direct inversion of $\mathbf{A}$, the operation count is thus reduced by a factor of more than 13[4].

Having solved system (3.12) for the correcting in-plane velocities, they are transformed back to world space according to

$$\Delta\mathbf{v}_i^{3D} = \mathbf{R}\Delta\mathbf{v}_i^{2D} . \tag{3.14}$$

The effect of these elemental corrections can be visualized by applying corresponding displacements $\Delta\mathbf{u}_i = h\Delta\mathbf{v}_i$ to individual elements in isolation. This results in an intermediate state with a set of incompatible triangles as shown in Fig. 3.4.

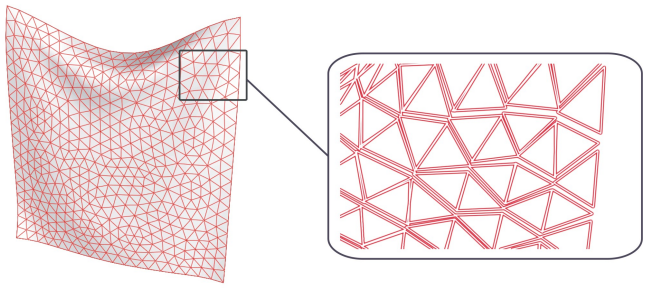


Figure 3.4: Effect of elemental corrections visualized by applying corresponding displacements to triangles in isolation.

Having established the formulation and enforcement of local deformation constraints, we can now turn to the combination of elemental responses into a global solution. But before proceeding to this subject in the next section, two further points should be mentioned.

[4]These numbers refer to optimized C++ code generated by the computer algebra software Maple.

First, we have only considered deformation constraints, but an analogous formulation can be used to enforce limits on strain rates. For this purpose, it is sufficient to replace the displacement vector $\mathbf{u}^e$ in Eq. (3.2) with nodal velocities $\mathbf{v}^e$ and to define corresponding strain rate limits.

Second, we have not considered the case of triangles with nodes subject to position constraints. This can be dealt with by modifying the matrix $\mathbf{A}$ of the deformation limiting problem accordingly. For the purpose of illustration, suppose that for a given element a single node with index $i = 3$ is fixed. In this case, we delete the fifth and sixth columns of $\mathbf{A}$ to obtain a modified matrix $\tilde{\mathbf{A}}$. The resulting overdetermined system is solved in a least squares sense using the normal equations

$$[\Delta\mathbf{v}_1 \; \Delta\mathbf{v}_2]^t = \left(\tilde{\mathbf{A}}^t\tilde{\mathbf{A}}\right)^{-1}\tilde{\mathbf{A}}^t\mathbf{b} \; . \tag{3.15}$$

3.3.2 Iterative Global Enforcement

The previous section introduced deformation constraints on triangle elements and described how to enforce them locally. This section discusses two iterative schemes for combining local element responses into a global solution.

In order to provide the necessary context, we assume that $\mathbf{x}^0$ holds positions at the beginning of a given time step and use $\mathbf{x}^1$ to denote candidate positions for the end of the step, obtained by integrating the equations of motion forward in time. Additionally, we define the average velocity of the time step as $\mathbf{v}^0 = \frac{1}{h}(\mathbf{x}^1 - \mathbf{x}^0)$. We can now cast the global enforcement of deformation limits as the problem of finding correcting velocities $\Delta\mathbf{v}$ such that the final positions

$$\mathbf{x} = \mathbf{x}^0 + h(\mathbf{v}^0 + \Delta\mathbf{v}) \tag{3.16}$$

satisfy all constraints. The global correction vector $\Delta\mathbf{v}$ is computed iteratively from elemental contributions and we consider two alternatives for this purpose.

Jacobi Iterative Enforcement Each iteration of global constraint enforcement consists of a loop over all triangles in the mesh, involving the computation of correcting velocities $\Delta\mathbf{v}^e$ for those elements whose deformation exceeds the limits. These corrections are accumulated at the vertices of the mesh and each vertex i stores contributions from all elements j incident to it as

$$\Delta\mathbf{v}_i = \sum_j \Delta\mathbf{v}_i^j \; . \tag{3.17}$$

For implementational convenience, we directly update candidate velocities and positions after iteration k as

$$\mathbf{v}^{k+1} = \mathbf{v}^k + \Delta\mathbf{v} \tag{3.18}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k + h\Delta\mathbf{v} \tag{3.19}$$

and reset $\Delta\mathbf{v}$ after each pass instead of accumulating corrections over the entire enforcement step. The iteration terminates when the deformation constraints are satisfied for all elements. In analogy to the iterative solution of linear systems of

equations, this process can be considered as *Jacobi* iterative enforcement [BFA02], since element responses of one iteration become visible to other elements only in the next iteration. One advantage of this approach is that the result is completely independent of element ordering in the mesh. Another benefit is that the elemental corrections are independent of each other and can thus be carried out in parallel.

Gauss-Seidel Iterative Enforcement Instead of computing all element corrections in isolation before applying them globally, an alternative approach is to apply them immediately. To this end, we initialize $\mathbf{x}^{k+1} = \mathbf{x}^k$ at the beginning of iteration $k+1$ before starting the loop over the elements. Having computed the velocity correction $\Delta\mathbf{v}^e$ for a given element, the positions of its three nodes are updated according to

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^{k+1} + h\Delta\mathbf{v}_i^e \; . \tag{3.20}$$

In this way, the response for an element is immediately visible to its neighbors and is taken into account when computing further corrections in the same iteration. The resulting scheme can be considered as *Gauss-Seidel* iteration and it has been reported to converge faster than its Jacobi counterpart in the context of edge-based strain limiting [BFA02]. This improved performance comes, however, at the cost of losing order-independence, which may introduce bias into the results. Fortunately, this dependence can be largely eliminated by randomizing the order of element updates in each iteration and, using this strategy, we did not observe bias or other artifacts in practice.

Having introduced these iterative enforcement schemes, the question now arises as to which of them should be preferred in practice. We will postpone a detailed performance comparison of the methods to Sec. 3.4 and instead continue with an overview of their implementation.

Implementation Algorithm 3.1 describes the implementation of iterative continuum-based strain limiting. It is structured into two loops: the inner one (ll.4-17) computes corrections for triangle elements in isolation, whereas the outer one (ll.2-22) combines these elemental responses and stops the process when all constraints are satisfied. While this is a straightforward implementation, its efficiency can be improved using the following modification.

In each iteration of the algorithm, all elements of the mesh are checked for constraint violations and corrections are computed and applied if necessary. Depending on the animation, however, the fraction of actually corrected elements can be rather small and is often less than 10%. In order to not waste time on evaluating constraints for elements that do not require correction, we pursue the following modified strategy. The first iteration proceeds as usual, but we mark all elements that needed correction. After this first iteration, we insert all marked elements as well as their incident triangles[5] into an index set, to which we refer as the *active set*. The underlying reasoning is that correcting a given element can only induce secondary deformations for triangles incident to it. The active set therefore contains all triangles that can possibly violate the limits and it is sufficient to check only these elements in the next iteration. The same process is applied in subsequent

[5]Two triangles are incident if they have at least one vertex in common.

Algorithm 3.1 limitDeformations($\mathbf{x}$,$\mathbf{v}$)

```
1: converged = false;
2: while not converged do
3:    converged = true;
4:    for (e = 1 to e = n_e) do
5:       x^e = getElementVec(e,x);          //get candidate positions for element
6:       ε = computeDeformation(x^e);
7:       if limitsViolated(ε) then
8:          converged = false;
9:          Δv^e = computeCorrection(ε);
10:         if Gauss-Seidel then
11:            addElementVec(e,Δv^e, v);  //commit element response immediately
12:            addElementVec(e,hΔv^e, x);  //commit element response immediately
13:         else
14:            addElementVec(e,Δv^e, Δv); //Jacobi iteration, store element response
15:         end if
16:      end if
17:   end for
18:   if not Gauss-Seidel then
19:      v = v + Δv;
20:      x = x + hΔv;
21:   end if
22: end while
```

iterations and we keep elements that were added to the set for the entire global enforcement step such that its size can only increase.

The implementation of this active set method requires only minimal changes to Alg. 3.1. When used in combination with Jacobi-type global enforcement, the number of necessary iterations is the same as before, but typically with much less elemental constraint evaluations. Compared to conventional Gauss-Seidel iteration, the speed at which isolated deformations are spread out can be slightly lower. This is due to the fact that, for a given iteration, the active set can only grow by the elements contained in its border, i.e., triangles that are incident to those already in the set. However, the increase in iteration count is only small whereas the gain in computational efficiency can be significant, as will be shown in Sec. 3.4.

Integration Algorithm 3.1 can be interpreted as a post-integration filter that modifies the average velocity of a time step such that the final positions satisfy all constraints. The same principle is used in the collision handling framework of Bridson et al. [BFA02]. Consequently, CSL can be used as a drop-in replacement for conventional edge-based strain and strain rate limiting methods. This integration is summarized in Algorithm 3.2. A time step of unconstrained physics provides candidate positions and velocities (l.3), which are fed to CSL for constraint enforcement (l.5). The corrected positions and velocities are then passed on to the collision handling scheme (l.6). This yields collision resolving velocities as well as final nodal positions. Another CSL pass can be applied to the final velocities in order to limit the strain rate (l.8).

Algorithm 3.2 CSL - Integration

```
1: //Simulation loop:
2: for (n = 1 to n = n_steps) do
3:    (x^{n+1}, v^{n+1}) = stepPhysics(h, x^n, v^n);
4:    limitStrain(x^{n+1}, v^{n+1});
5:    handleCollisions(x^n, x^{n+1}, v^{n+1});
6:    limitStrainRate(v^{n+1});
7: end for
```

3.3.3 Selecting Deformation Limits

We have so far not discussed how deformation limits can or should be selected in order to obtain a desired material behavior. There are essentially two ways that can be envisaged.

One is to let the animator or artist specify two parameters per deformation mode, corresponding to a stiffness coefficient for the elastic range and a deformation limit. This is a concise and intuitive interface that allows easier control of material behavior than the manipulation of nonlinear stress-strain curves. If desired, more complexity could be hidden from the user by quantizing the elastic stiffness into a small set of material behaviors, e.g, ranging from *'very soft'* to *'stiff'*.

However, it is also possible to determine these six parameters in order to match a given material behavior as closely as possible. In this context, it is insightful to take a closer look at the material curves shown in Fig. 3.5.

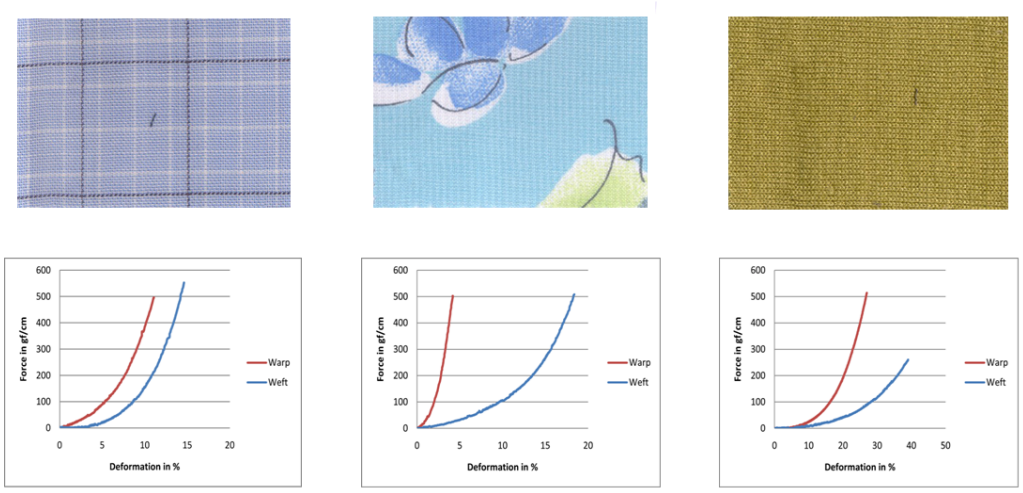


Figure 3.5: Photographs of real fabric samples (*top row*) and corresponding strain-stress plots (*bottom row*) for stretching in weft and warp directions. The first column shows a plain weave material made of wool and viscose. The second fabric is a light viscose-polyester composite in plain weave, while the third sample is a stretchy knit fabric with wool-viscose yarns.

The data was acquired in the context of the Virtual Try-On project [WKK+04] using the Kawabata Evaluation System [Kaw80a]. Accordingly, dimensionless strain is plotted against applied loading in gram force per centimeter (gf/cm).

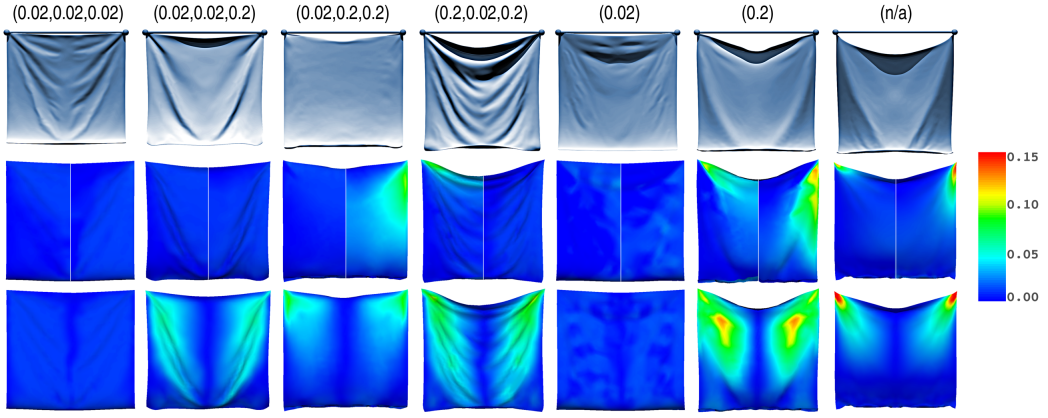


Figure 3.6: Representative frame from Example 3.1, which compares continuum- (columns 1–4) and edge-based (columns 5–6) strain limiting using different deformation constraints as indicated above each column. Results obtained without strain limiting are included as well (column 7). The top row shows rendered views, the middle row shows visualizations of weft and warp deformation in split images while the last row plots shear deformation. Weft and warp directions coincide with the horizontal and vertical axis of the image plane, respectively.

For a maximum loading of 500 gf/cm the resulting maximum deformation varies from 15% to 40%. However, it should be noted that a force of 500 gf/cm corresponds to loading a fabric sample of 1m length with 50 kg or to grabbing a part of it by hand and pulling with the equivalent of around 5 kg. Such high loadings are not to be expected under normal circumstances and it seems reasonable to set up deformation limits corresponding to forces roughly between 100 and 200 gf/cm. For the fabrics shown in Fig. 3.5 reasonable choices would be 5%, 2%, and 15% of admissible deformation in warp direction. Having estimated the deformation limit, the elastic stiffness coefficient can be fit to best approximate the remaining part of the stress-strain curve.

3.4 Results

We have investigated the performance of continuum-based deformation limiting on a number of examples, some of which are presented in this section. All computation times that are given in the following were obtained on a commodity workstation with 2GB of main memory and two dual-core AMD Opteron CPUs running at 2.0GHz. Unless indicated otherwise, only a single core was used.

The first example is meant to demonstrate the range of material behaviors that can be obtained with continuum-based strain limiting by merely varying the anisotropic deformation limits. For comparison, results for conventional edge-based strain limiting (ESL) are included as well. The setup consists of a square, irregularly tessellated mesh with 3616 faces, which is pinned at two corners and swings under the influence of gravity. We impose deformation constraints with a strictness varying from 0.02 to 0.2. Fig. 3.6 shows a rendered frame along with strain distributions for the cases studied. Only a single constraint value can be

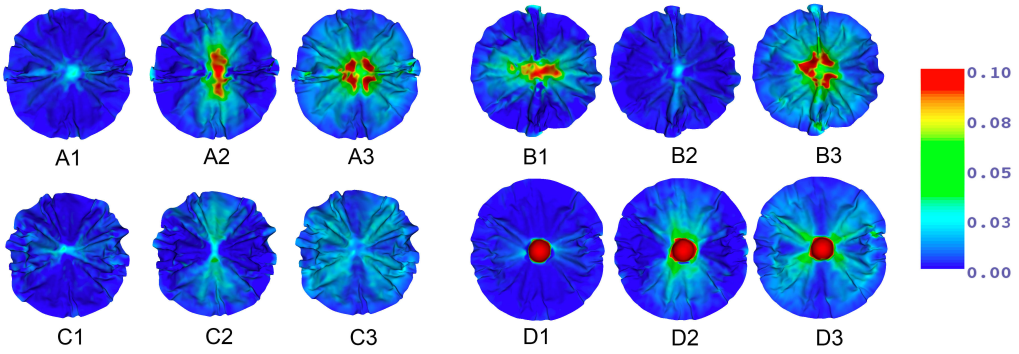


Figure 3.7: Strain plots for weft, warp and shear strain (left to right) in groups of three for Example 3.2. Material coefficients and deformation limits are set to (1000,200,100) N/m and (0.04,0.2,0.4), respectively. The plots show results obtained with CSL (A1-A3), CSL with material coefficients and deformation limits switched along weft and warp directions (B1-B3), ESL (C1-C3), and without strain limiting (D1-D3).

used for ESL (columns 5–6), but with CSL we can additionally experiment with differently strict constraints for each deformation mode (columns 1–4). In order to best isolate the effect of strain limiting, material parameters were set to (100,100,30) N/m in all cases[6]. Using CSL, substantially different material behaviors are obtained just by switching anisotropic constraints along weft and warp direction (compare columns 3 and 4). Likewise, setting strict constraints for weft and warp but a weak constraint for shear deformation gives completely different though realistic results. Relaxing the single constraint for ESL from 0.02 to 0.2 (columns 5 and 6), the fabric becomes globally softer and resembles the unconstrained case (column 7). The strain visualizations in rows 2 and 3 reveal that in all cases deformation constraints are accurately adhered to. The plots also show that CSL achieves a clean separation between the different deformation modes. In particular, we can observe higher deformation for strain components with softer constraints. We further note that the strain distribution for ESL is not as smooth as for CSL. This can be attributed to the fact that some edges are well aligned with the material directions while others are not.

The impression of this first experiment is largely confirmed by Example 3.2, which consists of a disc-shaped mesh (4424 faces, 1m diameter) that drapes over a small sphere. In contrast to the first example, we used a stiffer and strongly anisotropic material with elasticity coefficients and deformation limits selected in equal proportions as (1000,200,100) N/m and (0.04,0.2,0.4), respectively. The single parameter for edge-based strain limiting was set to 0.04. Figure 3.7 shows strain plots for a representative frame of this animation. Using continuum-based strain limiting, the deformations adhere accurately to the strict limit imposed in weft direction (A1). At the same time, larger strains are produced in warp and shear directions (A2-A3), which is in accordance to the more generous limits set

[6]Triples of elasticity coefficients or deformation limits refer to stretching in weft and warp direction and shearing, respectively.

for these modes. As can be seen from plots (B1-B3), a consistent behavior is obtained when switching the material coefficients and deformation limits in weft and warp directions. Using edge-based strain limiting, the threshold is respected, but in contrary to the material properties, deformation is more or less isotropic (C1-C3). The fact that deformation limiting is actually necessary to satisfy the imposed thresholds can be observed in plots (D1-D3), for which no strain limiting was used. The results show strongly localized deformations which exceed the imposed limits by a factor of up to 5.

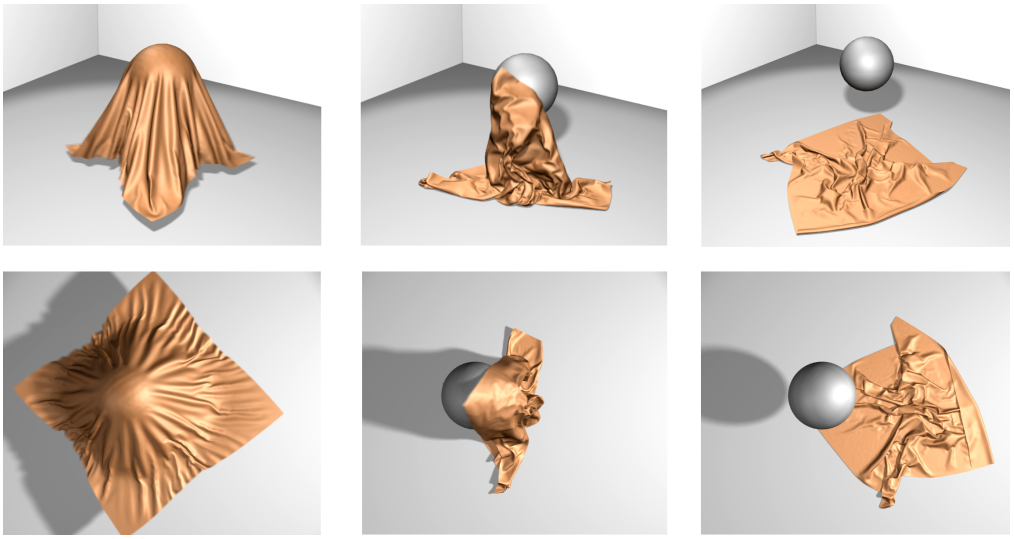


Figure 3.8: Three representative frames from Example 3.3 shown in side view (*top row*) and from above (*bottom row*).

The remaining examples are intended to demonstrate the method's ability to produce more complex and dynamic cloth animations. Example 3.3 consists of a square piece of cloth (14 476 triangles) with 0.5m side length that is placed slightly off the center of a small sphere with 0.15m radius. The material parameters were set to (300,300,150) N/m and deformation limits were chosen as (0.1,0.1,0.4). During three seconds of animation, the cloth first drapes around the sphere but subsequently slides down onto the floor, forming complex folding and buckling patterns as shown in Fig. 3.8. In this example, the average computation time for constraint enforcement was 2.92 seconds for a single 25Hz frame, while time integration took 7.8 seconds. As a side note, more than 55% of the total time was spent on collision handling, which can be attributed to the complex self collisions that occur throughout the animation. We also used this example to evaluate the efficiency of the modified iteration strategy[7] described in Sec. 3.3.2. Using this active set method, the average iteration count increased from 3.4 to 4.0, but the time spent on strain limiting decreased by almost 50% to 1.53 seconds per frame. This is an appreciable improvement and we found it to be consistent for all examples considered.

[7] In order to allow better comparison with edge-based strain limiting, the active set method was only used for Example 3.3.

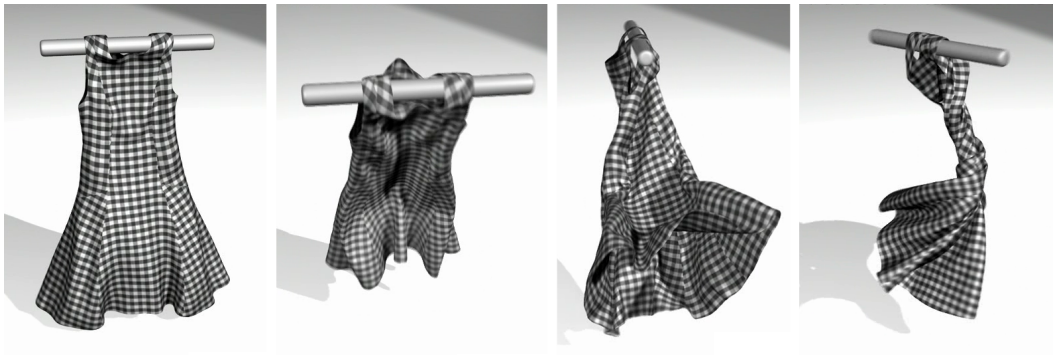


Figure 3.9: Four representative frames from a dress animation (Example 3.4).

The fourth example is a practical cloth animation consisting of a dress (8891 triangles) and a rigid bar with scripted motion that is placed through its straps. The elastic material coefficients are set to (300,300,75) N/m and deformation limits are selected as (0.1,0.1,0.4), providing room for large shear deformations. During five seconds of animation, the dress initially drapes due to gravity, but is soon accelerated forwards by the abrupt motion of the bar, which moves back and forth several times. This leads to high-velocity collisions which are challenging for both strain limiting and collision handling. Subsequently, the bar is rotated with increasing angular velocity such as to spin and swirl the dress around itself (see Fig. 3.9). Complicated self collisions result in multiple twisted layers and provoke large shear deformations.

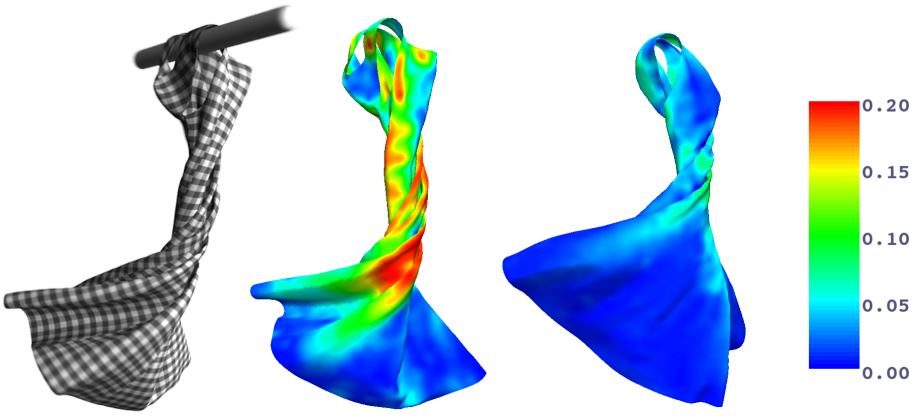


Figure 3.10: Rendered frame (*left*) and shear strain plots using CSL (*middle*) and ESL (*right*). With CSL, larger shear strains can be allowed while enforcing low stretch deformation. ESL can only enforce a single limit on stretch and shear deformation, leading to substantially different results.

Again, the difference between continuum- and edge-based deformation limiting can be seen very clearly in this example. The edge-based scheme cannot satisfy stretch constraints and allow large shear deformations at the same time (see Fig. 3.10, right). Using CSL, larger shear deformations develop in accordance to the prescribed limits and the elastic material (Fig. 3.10, middle).

For this example, time integration took about 16.09 minutes for the five seconds of animation. The average iteration count for CSL was 14.92 and the corresponding computation time was 12.19 minutes.

Finally, a last example is used to compare the computational costs and convergence properties of CSL and ESL. The setup for this experiment is the same as in Example 3.1, but a set of meshes with increasing resolution is used. In order to facilitate comparison to the edge-based variant, an isotropic material with material coefficients of (500,500,250) N/m was used and deformation limits were set to (0.1,0.1,0.2) for CSL and 0.1 for ESL. Table 3.1 provides average iteration counts and run times for 1 second of simulation.

	226 faces			904 faces			3616 faces		
	#it	t_sl	t_int	#it	t_sl	t_int	#it	t_sl	t_int
CSL-GS	1.00	0.34	3.24	1.04	1.39	14.14	3.81	18.98	77.29
CSL-JAC-1	1.00	0.47	3.32	1.04	1.85	14.11	5.45	36.90	73.20
CSL-JAC-4	1.00	0.26	3.51	1.04	0.90	14.04	5.45	15.88	76.01
ESL	1.00	0.13	3.35	1.05	0.62	13.99	3.37	4.87	73.85

Table 3.1: Iteration counts and computation times in seconds for Example 3.5. *#it* denotes the average number of iterations to convergence for strain limiting and *t_sl* the corresponding computation times. *t_int* lists time spent on time integration.

It can be seen that the Gauss-Seidel variant (CSL-GS) converges slightly faster than its Jacobi counterpart (CSL-JAC-1), which also translates into faster computations for the single-threaded case. However, using 4 threads on 4 CPUs, the parallel Jacobi variant (CSL-JAC-4) is faster than CSL-GS. As expected, the computational costs for the CSL variants are higher than for ESL, but this difference has to be weighted against the increased capabilities offered by CSL.

3.5 Conclusions

Summary This chapter presented continuum-based strain limiting, which is a novel approach for simulating biphasic, anisotropic textiles. Deformation constraints were derived from a continuous strain measure and discrete expressions were obtained using linear triangle finite elements. Our method allows accurate control over all deformation modes and enables the use of individual anisotropic thresholds. Unlike conventional edge-based approaches, CSL does not require specifically-aligned discretizations but works on unstructured triangle meshes, which is a valuable asset when simulating complex garments.

The capability of continuum-based strain limiting to create diverse material behaviors has been demonstrated on a number of examples. It was also shown that conventional edge-based approaches can only reproduce a subset of the effects achievable with CSL.

Limitations and Future Directions Continuum-based strain limiting relies on iterative constraint enforcement, as did previous edge-based methods described in

[Pro95, BFA02]. As such, it inherits the drawbacks with respect to performance scaling with increasing mesh sizes, discussed in [GHF+07]. An interesting direction for future work would therefore be to explore ways to improve the asymptotic convergence behavior, e.g., using multi-resolution techniques or direct enforcement schemes. The fast projection scheme of Goldenthal et al. [GHF+07] seems an attractive approach in this context. However, a fundamental problem is the fact that imposing deformation constraints on a per-triangle basis leads to far too many constraints for the degrees of freedom in the system. English et al. [EB08] avoid this problem by using a discontinuous discretization. Indeed, our continuum-based deformation constraints can be used as a drop-in replacement for their edge length constraints. However, their method introduces a significant amount of additional degrees of freedom, leading to higher computational costs.

Another point for improvement is the interplay between deformation limiting and collision handling. Implemented as a velocity filter, continuum-based deformation limiting integrates seamlessly into the widely used collision handling framework of Bridson et al. [BFA02]. This makes switching to CSL a lightweight effort, but a limitation of the original method persists: in order to obtain intersection-free configurations, the collision handling filter has to be applied last, i.e., after strain limiting. Collision response can therefore reintroduce deformations that were painstakingly removed before. Although strain rate limiting can reduce secondary deformations in the next time step, it cannot remove strain introduced by collision handling. An approach worth investigation might therefore be to couple the enforcement of deformation and collision constraints.

Finally, a further direction is to extend CSL to volumetric solids in order to model, e.g., living tissue. Especially tendons and muscles naturally exhibit anisotropic and often biphasic properties[8]. In this context, continuum-based deformation constraints could be used to build a computationally efficient approximation of complex and costly nonlinear material laws.

[8] see, for example, the textbook by Fung [Fun93]

Chapter 4

Asynchronous Cloth Simulation

Physically-based approaches for cloth animation give rise to stiff equations of motion, which render numerical time integration a challenging problem. Due to their superior stability properties implicit methods are typically preferred over their explicit counterparts. Especially the semi-implicit Euler scheme enjoys widespread use since it offers fast computation times, provided that large steps can be taken. This, however, is not always possible or even desirable. First, complex collisions typically require reduced step sizes in order to be resolved in a visually pleasing manner and this reduction lessens the computational advantages of implicit methods. Second, large step sizes and stiff materials are known to amplify the effect of numerical dissipation, leading to overdamped animations that lack detail (see Fig. 4.1, (*a*)).

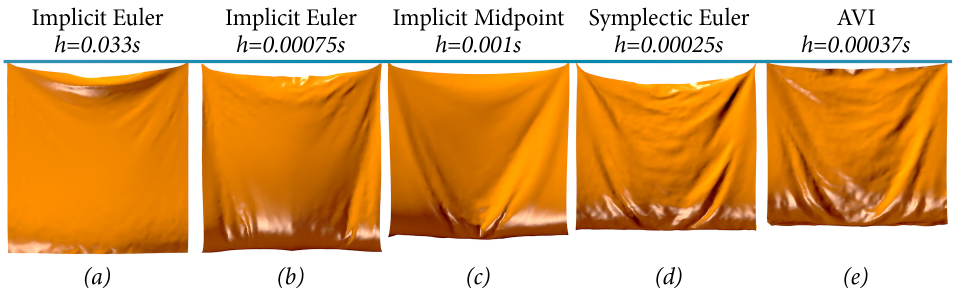


Figure 4.1: Comparison of different integration schemes on an animation of a swinging cloth with low material damping. Using the semi-implicit Euler method with a large step size (*a*) leads to a considerable lack of detail when compared to results for smaller step sizes (*b*). For the same computation time, our AVI-based method (*e*) produces more detail.

Explicit methods, by contrast, do not suffer from numerical dissipation but possess only limited stability when applied to stiff equations. For conventional methods[1], the step size has to be small enough to match the stability requirements of the stiffest component of the system. This is especially unfortunate when using unstructured meshes, where a few small elements, e.g., in regions around curved borders can drastically limit the global step size and thus computational efficiency.

[1] i.e., *synchronous* methods

Overview and Contributions Motivated by these observations, this chapter explores an alternative approach based on *asynchronous explicit time stepping*. The enabling technology of this method is the asynchronous variational integrator (AVI) introduced by Lew et al. [LMOW03]. Using an explicit integrator eliminates the problem of numerical dissipation such that damping becomes fully controllable through material parameters (see Fig. 4.1, (*e*)). The asynchronous formulation relaxes the strict limitations of synchronous methods by assigning each element an individual step size according to its local stability requirements.

In order to be of practical use for cloth animations, collision handling has to be integrated into the asynchronous framework. We propose a three-stage strategy for this purpose that combines both synchronous and asynchronous techniques. The efficiency and robustness of the solver is further increased through per-element deformation limiting and adaptive step size reduction.

The remainder of this chapter is organized as follows. In order to provide the necessary context, we first review the basic concepts of variational integration and briefly summarize the derivation of AVIs (Sec. 4.1). Details on the asynchronous time stepping framework are given in Sec. 4.2 and we describe our approach to collision handling in Sec. 4.3. Results and examples are presented in Sec. 4.4, followed by a conclusive summary in Sec. 4.5.

4.1 Background

The method described in this chapter is based on an asynchronous time stepping scheme, which has its roots in discrete mechanics. We start by summarizing the template process for constructing variational time integrators. Having introduced these basic concepts in the synchronous setting, we subsequently consider the translation to the asynchronous case. An extensive treatment of the underlying concepts from classical mechanics can be found, e.g., in the standard textbook by Arnold [Arn97]. For a comprehensive overview on discrete mechanics and variational integrators we refer to the survey article by Marsden and West [MW01].

Variational Integrators We consider a discrete mechanical system consisting of a set of n_v nodes whose configuration is described position and velocity vectors $\mathbf{x} \in \mathbb{R}^{3n_v}$ and $\dot{\mathbf{x}} \in \mathbb{R}^{3n_v}$, respectively. The system is characterized by its *Lagrangian* L, which is defined as the difference between kinetic and potential energy

$$L(\mathbf{x}, \dot{\mathbf{x}}) = T(\dot{\mathbf{x}}) - V(\mathbf{x}) \; . \tag{4.1}$$

In our case, V contains contributions from elastic and gravitational potential energy. The kinetic energy is defined as

$$T(\dot{\mathbf{x}}) = \frac{1}{2} \dot{\mathbf{x}}^t \mathbf{M} \dot{\mathbf{x}}, \tag{4.2}$$

where $\mathbf{M}$ is the (diagonal) mass matrix as described in Chapter 2. The motion of the system is described by Hamilton's principle, which states that the trajectory $\mathbf{x}(t)$ between two fixed configurations $\mathbf{x}(t^0)$ and $\mathbf{x}(t^N)$ is such that the action

$$S = \int_{t^0}^{t^N} L(\mathbf{x}(t), \dot{\mathbf{x}}(t)) \; dt \; , \tag{4.3}$$

is rendered stationary. This implies that

$$\delta S = \int_{t^0}^{t^N} \left[\frac{\partial L}{\partial \mathbf{x}} - \frac{d}{dt}\frac{\partial L}{\partial \dot{\mathbf{x}}}\right] \delta\mathbf{x} + \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\mathbf{x}}}\delta\mathbf{x}\right)\, dt = 0\,, \tag{4.4}$$

for all variations satisfying $\delta\mathbf{x}(t^0) = \delta\mathbf{x}(t^N) = 0$. The latter requirement also leads to the vanishing of the last term, which reveals the Euler-Lagrange equations as

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\mathbf{x}}} = \frac{\partial L}{\partial \mathbf{x}}\,. \tag{4.5}$$

Inserting the definition of the Lagrangian into this expression, we recover the ordinary differential equations (2.38) which are the starting point for *conventional* time discretization schemes such as those discussed in Chapter 2.

Instead of discretizing these equations of motion, *variational* integrators are derived by approximating the action integral with a discrete action sum

$$S_D = \sum_{n=0}^{N-1} L^{n+1}(\mathbf{x}^n, \mathbf{x}^{n+1}, t^n, t^{n+1}) \tag{4.6}$$

where the pairs $(\mathbf{x}^n, t^n)$ define the discrete trajectory of the system and

$$L^{n+1}(\mathbf{x}^n, \mathbf{x}^{n+1}, t^n, t^{n+1}) \approx \int_{t^n}^{t^{n+1}} L(\mathbf{x}(t), \dot{\mathbf{x}}(t))\, dt \tag{4.7}$$

is the discrete Lagrangian. A corresponding discrete version of Hamilton's principle [MW01] requires that the sequence $\mathbf{x}_1, \ldots, \mathbf{x}_{N-1}$ renders S_D stationary. This leads to the discrete Euler-Lagrange equations, which require

$$\frac{\partial L^n(\mathbf{x}^{n-1}, \mathbf{x}^n, t^{n-1}, t^n)}{\partial \mathbf{x}^n} + \frac{\partial L^{n+1}(\mathbf{x}^n, \mathbf{x}^{n+1}, t^n, t^{n+1})}{\partial \mathbf{x}^n} = 0 \tag{4.8}$$

to hold for all $n = 1, \ldots, N-1$. These equations give direct rise to a time stepping scheme once a discrete Lagrangian has been defined. In order to illustrate this process, we make a particular choice for approximating the integral in Eq. (4.7). To this end, we assume piecewise linear trajectories (i.e., constant velocities on $[t^n, t^{n+1}]$) and use a rectangle rule for quadrature that evaluates the potential energy at the end of the interval. This gives

$$L^{n+1}(\mathbf{x}^n, \mathbf{x}^{n+1}, t^n, t^{n+1}) = (t^{n+1} - t^n) L(\mathbf{x}^{n+1}, \frac{\mathbf{x}^{n+1} - \mathbf{x}^n}{t^{n+1} - t^n})\,. \tag{4.9}$$

Inserting this expression into the discrete Euler-Lagrange equations (4.8) yields

$$\mathbf{M}(\frac{\mathbf{x}^{n+1} - \mathbf{x}^n}{t^{n+1} - t^n} - \frac{\mathbf{x}^n - \mathbf{x}^{n-1}}{t^n - t^{n-1}}) = -(t^n - t^{n-1})\frac{\partial V(\mathbf{x}^n)}{\partial \mathbf{x}^n}\,, \tag{4.10}$$

which can be rearranged to give an update scheme for $\mathbf{x}^{n+1}$ in terms of $\mathbf{x}^n$ and $\mathbf{x}^{n-1}$. In the special case of a constant step size h , this expression simplifies to

$$\mathbf{M}(\mathbf{x}^{n+1} - 2\mathbf{x}^n + \mathbf{x}^{n-1}) = -h^2\frac{\partial V(\mathbf{x}^n)}{\partial \mathbf{x}^n}\,, \tag{4.11}$$

which can be identified as the two-step version of the explicit Verlet scheme (see Hairer et al. [HLW03]). In practice, a one-step formulation is more convenient and for this purpose, it is customary to introduce staggered[2] velocities as

$$\mathbf{v}^{n-\frac{1}{2}} = \frac{\mathbf{x}^n - \mathbf{x}^{n-1}}{h} \qquad \text{and accordingly} \qquad \frac{\mathbf{v}^{n+\frac{1}{2}} - \mathbf{v}^{n-\frac{1}{2}}}{h} = \mathbf{M}^{-1}\frac{\partial V(\mathbf{x}^n)}{\partial \mathbf{x}^n} \,. \quad (4.12)$$

The one-step update scheme follows as

$$\mathbf{x}^n = \mathbf{x}^{n-1} + h\mathbf{v}^{n-\frac{1}{2}} \quad (4.13)$$

$$\mathbf{v}^{n+\frac{1}{2}} = \mathbf{v}^{n-\frac{1}{2}} + h\mathbf{M}^{-1}\frac{\partial V(\mathbf{x}^n)}{\partial \mathbf{x}^n} \,. \quad (4.14)$$

This scheme allows the following interpretation: first, positions are updated with the constant velocity of the preceding interval. Second, velocities for the next interval are computed by applying instantaneous impulses to $\mathbf{v}^{n-\frac{1}{2}}$ according to the current potential. This pattern can also be found in the explicit AVI scheme, which is introduced subsequently.

Asynchronous Setting The previous paragraph demonstrated the construction of variational integrators for discrete mechanical systems in a general setting. In particular, no assumptions were made on the origin of the discretization or whether the discrete problem derives from a continuous one. We will now assume that V is an elastic potential defined by Eqs (2.19) or (2.21) and that the positions $\mathbf{x}$ stem from a finite element discretization of the corresponding continuum mechanics problem as described in Chapter 2. In order to proceed towards asynchronous integration, we will partition the Lagrangian in accordance to the underlying triangle mesh. To this end, we write its kinetic and potential components as

$$V = \sum_{K\in\mathcal{K}} V_K \qquad \text{and} \qquad T = \sum_{K\in\mathcal{K}} T_K \,, \quad (4.15)$$

where V_K and T_K correspond to the potential and kinetic energies of element K and $\mathcal{K}$ denotes the set of triangle elements. The discrete elastic energy V_K is only a function of the discrete deformation gradient on element K, which in turn is given by Eq. (2.31). For an isotropic material law according to Eq. (2.21), we simply have

$$V_K = \frac{1}{2}\lambda \mathrm{tr}(\mathbf{E}_K)^2 + \mu(\mathbf{E}_K : \mathbf{E}_K), \qquad \text{where} \qquad \mathbf{E}_K = \frac{1}{2}(\mathbf{F}_K^t\mathbf{F}_K - \mathbf{I}) \,. \quad (4.16)$$

An analogous expression holds for anisotropic materials. Finite elements can be used for the kinetic energy as well, but in practice we use a lumping scheme that distributes the mass of an element evenly among its vertices. To this end, we define the mass of node a due to element K as $m_{K,a} = \frac{1}{3}\rho A_K$, where A_K is the element's area and ρ denotes the mass per unit surface. The kinetic energy of element K is then obtained as

$$T_K = \frac{1}{2}\sum_{a\in K} m_{K,a}||\dot{\mathbf{x}}_a||^2 \,, \quad (4.17)$$

[2]This means that velocities are not co-located in time with positions but are associated with the intervals $[t^{i-1}, t^i]$.

which enables us to express the Lagrangian as a sum of elemental contributions,

$$L = \sum_{K \in \mathcal{K}} L_K = \sum_{K \in \mathcal{K}} (T_K - V_K) \, . \tag{4.18}$$

Letting $\mathbf{x}_K$ and $\dot{\mathbf{x}}_K$ denote vectors holding positions respectively velocities of element K, we can rewrite the action integral as

$$S = \int_{t^0}^{t^N} \sum_{K \in \mathcal{K}} L_K(\mathbf{x}_K(t), \dot{\mathbf{x}}_K(t)) \, dt = \sum_{K \in \mathcal{K}} \int_{t^0}^{t^N} L_K(\mathbf{x}_K(t), \dot{\mathbf{x}}_K(t)) \, dt = 0 \, , \tag{4.19}$$

which indicates the possibility to use different time discretizations of $[t^0, t^N]$ for the elemental Lagrangians L_K. Assigning each element its individual step size h_K yields sets of time events $(t_K^0, \ldots, t_K^{N_K})$, where $t_K^j = t^0 + j \cdot h_K$. The time sets of all elements incident to a given vertex a furthermore induce a nodal time set $(t_a^0, ..., t_a^{N_a})$, where each t_a^i is part of the time set of (at least) one of the neighbor elements of a.

In order to arrive at a time stepping method, the discrete action sum has to be constructed and, to this end, it remains to define the discrete Lagrangians of the triangle elements. Integrating the potential term V_K with the same quadrature rule as in (4.9) yields a semi-discrete Lagrangian L_K^j for the interval $[t_K^j, t_K^{j+1}]$ as

$$L_K^j = \int_{t_K^j}^{t_K^{j+1}} T_K(\dot{\mathbf{x}}_K(t)) \, dt + (t_K^{j+1} - t_K^j) V_K(\mathbf{x}_K^{j+1}) \, . \tag{4.20}$$

Unlike the potential energy, the integral of T_K cannot be evaluated in the same way as in (4.9), since the velocities $\dot{\mathbf{x}}_K(t)$ are no longer constant over $[t_K^j, t_K^{j+1}]$. The reason for this is that update events of neighboring elements cause changes to the nodes of K and their velocities during the interval $[t_K^j, t_K^{j+1}]$. In order to evaluate the integral of T_K explicitly, all these changes have to be taken into account. To this end, let n_j denote the number of update events for a given node a of K during a given interval $[t_K^j, t_K^{j+1}]$, which is thus subdivided as $[t_K^j = t_a^{j,0}, \ldots, t_a^{j,(n_j+1)} = t_K^{j+1}]$. Assuming piecewise linear trajectories, the integral of the kinetic energy $T_{K,a}$ of node a due to element K can now be written explicitly as

$$\int_{t_K^j}^{t_K^{j+1}} T_{K,a} = \frac{1}{2} m_{K,a} \sum_{i=0}^{n_j} \frac{||\mathbf{x}_a^{i+1} - \mathbf{x}_a^i||^2}{(t_a^{i+1} - t_a^i)} \, , \tag{4.21}$$

and summing over all nodal contributions yields a corresponding expression for the entire element. Further note that for every term in the sum of (4.21), there are contributions from all neighboring elements of node a, which are identical except for the mass factors $m_{K,a}$. Since we have $\sum_K m_{K,a} = m_a$, these terms merge into a single one with mass factor m_a. We can thus write the discrete action sum as

$$S_D = \sum_{K \in \mathcal{K}} \sum_{j=0}^{N_K - 1} L_K^j = \sum_{a=1}^{n_v} \sum_{i=0}^{N_a - 1} \frac{1}{2} m_a \frac{||\mathbf{x}_a^{i+1} - \mathbf{x}_a^i||^2}{(t_a^{i+1} - t_a^i)} - \sum_{K \in \mathcal{K}} \sum_{j=0}^{N_K - 1} (t_K^{j+1} - t_K^j) V_K(\mathbf{x}_K^{j+1}) \, .$$

The asynchronous update scheme is obtained from this expression by invoking the discrete Hamilton's principle [MW01], which requires that

$$\frac{\partial S_D}{\partial \mathbf{x}_a^i} = 0 \qquad \text{for all } a \in \{1, \ldots, n_v\} \text{ and all } i \in \{1, \ldots, N_a - 1\} \; .$$

Observe that a given $\mathbf{x}_a^i$ contributes to three terms of the discrete action sum: two kinetic terms corresponding to the contiguous intervals $[t_a^{i-1}, t_a^i]$ and $[t_a^i, t_a^{i+1}]$, respectively, as well as a potential component evaluated at time t_a^i. Carrying out the derivatives, we obtain

$$\frac{\mathbf{x}_a^i - \mathbf{x}_a^{i-1}}{t_a^i - t_a^{i-1}} - \frac{\mathbf{x}_a^{i+1} - \mathbf{x}_a^i}{t_a^{i+1} - t_a^i} = \frac{1}{m_a}(t_K^j - t_K^{j-1}) \frac{\partial V_K(\mathbf{x}_K^i)}{\partial \mathbf{x}_a^i} \; , \tag{4.22}$$

where we have used $t_a^i = t_K^j$. This is a two-step update scheme and in analogy to the synchronous case of (4.12), it can be transformed to a one-step method using staggered velocities. By doing so, we finally arrive at

$$\mathbf{x}_a^i = \mathbf{x}_a^{i-1} + (t_a^i - t_a^{i-1}) \mathbf{v}^{i-\frac{1}{2}} \tag{4.23}$$

$$\mathbf{v}_a^{i+\frac{1}{2}} = \mathbf{v}_a^{i-\frac{1}{2}} + \frac{1}{m_a}(t_K^j - t_K^{j-1}) \frac{\partial V_K(\mathbf{x}_K^i)}{\partial \mathbf{x}_a^i} \; . \tag{4.24}$$

These equations describe an asynchronous time stepping scheme, which is steered by asynchronous elemental update events at times t_K^j. Its algorithmic implementation as well as the extensions and adaptations made for the work presented in this chapter will be described in the next section.

4.2 Asynchronous Time Stepping

A high-level overview of the time stepping scheme is provided by Algorithm 4.1, which is structured into two main loops. The outer one (ll.7-30) is driven by the collision handling scheme, which synchronizes the system at regular instants t_c^i, separated by a collision step size Δt_c. The inner loop (ll.14-28) advances elements asynchronously across the intervals $[t_c^i, t_c^i + \Delta t_c]$. We postpone details on collision handling to the next section and first describe the asynchronous event loop.

Algorithm Overview The most distinguishing aspect of AVIs is that they allow each element to have its dedicated time step. Since each element advances at its own pace, also the nodes of the system evolve asynchronously as illustrated in Fig. 4.2. It is therefore necessary to keep track of both elemental and nodal times, denoted by t_K and t_a in Algorithm 4.1.

The inner loop of the latter corresponds to the update scheme of Lew et al. [LMOW03], modified in order to accommodate the extensions described in this chapter. It advances the system between two successive synchronization points t_c^i and $t_c^i + \Delta t_c$, at which global collision handling routines are performed (see Sec. 4.3). The event loop is implemented using a priority queue, which orders elemental update events automatically according to their time stamp. This is a

Algorithm 4.1 Outline of the time stepping algorithm.

```
1: //Initialization:
2: t_c = 0
3: for k = 1 to n_e do
4:    compute elemental time step h_k
5: end for
6: //Outer loop
7: while t_c < t_end do
8:    collisionHandling_stage1()
9:    for k = 1 to n_e do
10:      queue.push(k; t_c + h_k)                          //fill queue
11:   end for
12:   t_c = t_c + Δt_c
13:   //Asynchronous loop
14:   while queue is not empty do
15:      (k; t_K) ⇐ queue.pop()                           //retrieve next event
16:      K = element(k)                                   //get element
17:      for all a ∈ K do
18:         x_a = x_a + v_a(t_K − t_a)                    //update positions
19:         t_a = t_K                                     //update nodal times
20:         v_a = v_a − (h_K/m_a) f_{K,a}                 //update velocities
21:      end for
22:      limitStrainAndStrainRate()
23:      collisionHandling_stage2()
24:      if (t_K + h_K) ≤ t_c then
25:         t_K = t_K + h_K
26:         queue.push(K, t_K)                            //reschedule element
27:      end if
28:   end while
29:   collisionHandling_stage3()
30: end while
```

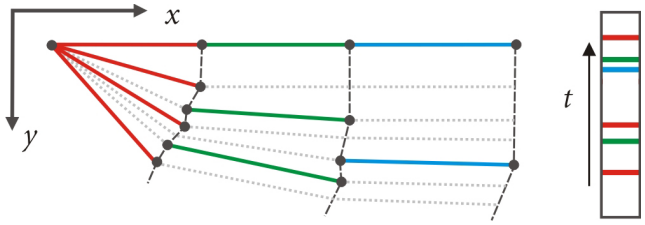


Figure 4.2: A chain is fixed at one point and subjected to gravity. *Left*: three elements with different step sizes advance asynchronously in time. The nodes follow piecewise linear trajectories. *Right*: element activations over time are indicated by corresponding color marks.

convenient and efficient way to ensure causality, since the element which has to be updated next is always on top of the queue.

The update procedure itself is as follows. Once an element is retrieved from the queue, its nodal positions are updated using the current velocities (l.18) and nodal times are set to the current element time (l.19). Subsequently, the nodal velocities are updated according to the net forces[3] that have been acting on the element since its last update (l.20). Finally, if the element's next update time lies in the interval $[t_c^i, t_c^i + \Delta t_c]$, it is rescheduled for evaluation (l.26). The event loop terminates once the queue is empty, which means that the next synchronization point for global collision handling has been reached.

Geometry output for rendering is generated at regular intervals. We use dedicated events for this purpose that extrapolate the system's positions to the corresponding point in time, using its current velocities.

Cloth Mechanics In order to simulate the mechanics of cloth within the asynchronous framework, the template expression in l.20 has to be specialized to appropriate elastic and viscous forces. Integrating elastic membrane forces is straightforward and the finite element expression (2.33) from Chapter 2 can be used without modification.

Similarly, dissipation is implemented as a viscous stress according to Eq. (2.25), which involves the rate of deformation tensor $\dot{\mathbf{F}}$. The obvious way to compute this quantity is to use the current nodal velocities, but a subtlety arises in this context: the velocities at an element's update time t_K^j are generally different from the average values according to the positions at times t_K^{j-1} and t_K^j[4], which is due to the impulses imparted by neighboring elements during the interval $[t_K^{j-1}, t_K^j]$. The dissipation impulses applied at time t_K^j should, however, approximate the integral of the corresponding forces over the preceding interval. A better reflection of the average behavior is obtained by using a difference approximation of the tensor as suggested by Lew [Lew03],

$$\dot{\mathbf{F}}_K^j = (t_K^j - t_K^{j-1})^{-1}(\mathbf{F}_K^j - \mathbf{F}_K^{j-1}) \,, \tag{4.25}$$

[3] We use the shorthand $\mathbf{f}_{K,a}$ to summarize all elastic, viscous and external forces due to element K that act on its node a.

[4] See also the corresponding discussion in [HVS+09].

where $\mathbf{F}_K^j$ and $\mathbf{F}_K^{j-1}$ are deformation gradients for the current and previous element updates.

With the elastic and viscous in-plane forces defined, it remains to integrate bending forces into the asynchronous framework and the hinge-based model [BMF03, GHDS03] is again an attractive candidate. A direct way of implementation is to use dedicated bending elements (consisting of two edge-adjacent triangles) and to add corresponding update events to the queue. However, the local collision handling routine has to be invoked whenever an element (membrane or bending) is updated (see Alg. 4.1, l. 23) such that maintaining a single type of element is desirable. We therefore decided to model bending as an external force that is applied to the nodes of a membrane element whenever it is updated. In order to compute bending forces on a given membrane element, we evaluate the three hinges corresponding to its edges, using the expression provided by Bridson et al. [BMF03].

As illustrated in Fig. 4.3, the support of these forces extends to four triangles and six nodes, three of which belong to neighbors of the current element (depicted in blue). Since these neighbor triangles are generally not synchronized with the current element, their positions have to be extrapolated accordingly before forces are computed.

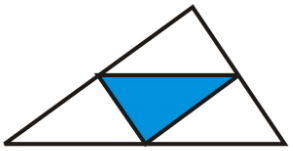


Figure 4.3: Support for bending forces.

Having coupled the computations of membrane and bending forces in this way, a step size that allows stable integration has to be determined. We do this based on the assumption that bending and damping forces are not critical for stability, such that the latter is determined solely by elastic properties. The critical step size, beyond which stability is jeopardized, is then determined by the Courant-Friedrichs-Lewy (CFL) condition (see, e.g., [Bat96]). Roughly speaking, this condition requires that the elemental time step be less than the time it takes a material wave to pass the element. Following Lew [Lew03], we set the step size h_K to a fraction of the critical value,

$$h_K = \alpha \, r_i \sqrt{\rho / (\lambda + 2\mu)} \, , \tag{4.26}$$

where r_i is related to the size of element i (its internal radius) and α is a user-supplied parameter, typically chosen as $0 < \alpha < 1$. The Lamé constants λ and μ describe an isotropic, linear-elastic material and ρ denotes the mass density.

As can be seen from Eq. (4.26), reducing the material stiffness leads to larger elemental time steps. This allows for faster integration but it also fits well with the biphasic model of cloth as described in the previous chapter. To implement this model in the context of asynchronous time stepping, we complement elastic forces with local deformation limiting as described next.

Monitoring and Limiting Deformations The robustness and efficiency of the solver can be improved by monitoring and limiting element deformations at run time (see Alg. 4.1, l.22). In keeping with the original work [TPS08], the following exposition is based on edge deformations. The extension to the continuum-based method described in the previous chapter is, however, straightforward.

Assume that a given element K is processed in the course of asynchronous time stepping and that its positions and velocities have been updated according to ll.18 and 20 of Algorithm 4.1. Let $\mathbf{x}_1$ and $\mathbf{x}_2$ denote the position vectors for two of the element's vertices and let the corresponding edge have a rest length l_0 such that its current strain is

$$\varepsilon_c = \frac{||\mathbf{x}_2 - \mathbf{x}_1|| - l_0}{l_0} = \frac{l_c}{l_0} - 1 \; . \tag{4.27}$$

Using the current velocities of the edge points, $\mathbf{v}_1$ and $\mathbf{v}_2$, we define the approximate strain ε_n for the time of the next element update as

$$\varepsilon_n = \frac{||\mathbf{x}_2 + h_K\mathbf{v}_2 - (\mathbf{x}_1 + h_K\mathbf{v}_1)|| - l_0}{l_0} = \frac{l_n}{l_0} - 1 \; . \tag{4.28}$$

Additionally, a corresponding rate of deformation is defined as

$$\dot{\varepsilon} = (\varepsilon_n - \varepsilon_c)/h_K = (l_n - l_c)/(l_0 h_K) \; . \tag{4.29}$$

These three quantities are used to implement a strategy for monitoring and limiting deformations as follows. Whenever the predicted next strain ε_n exceeds a given threshold ε_n^{lim}, momentum-conserving impulses are computed such that, when applied to the current velocities, the condition $\varepsilon_n = \varepsilon_n^{lim}$ holds. To this end, we compute a length correction Δl_n that satisfies

$$\frac{(l_n + \Delta l_n) - l_0}{l_0} = \varepsilon_n^{lim} \rightarrow \Delta l_n = l_0\varepsilon_n^{lim} - (l_n - l_0) \; . \tag{4.30}$$

This correction is distributed to the two vertices such that their center of mass remains unchanged and is applied as equivalent impulses to the current velocities. This limiting scheme works well if the current deformation is below the threshold or if violations are only moderate. However, if the current deformation already exceeds the limit ε_n^{lim} by a significant amount, large impulses would be required to recover a configuration with valid strain. In order to improve stability in such situations, we additionally correct positions using a formulation analogous to (4.30). This is done whenever the condition $\varepsilon_c > \varepsilon_c^{lim}$ holds, where ε_c^{lim} is a second deformation threshold[5]. Note that, in contrast to conventional strain limiting, there is only a single correction per element update, i.e., no iterations are performed. Although this is not appropriate for strictly enforcing tight deformation limits, it is an effective way to prevent large strains and thus increase the robustness of the solver.

In order to further improve stability in critical situations, we also take advantage of the possibility to selectively reduce the step size of individual elements. In cases of sudden character motion or high velocity impacts, the collision-resolving impulses can induce large strain rates, which threaten to destabilize the simulation. Such situations can be identified on the basis of the current strain rate $\dot{\varepsilon}$ or, equivalently, the expected change in strain $\Delta\varepsilon = h_K\dot{\varepsilon}$ during the element's next time step. Following the suggestion of [BFA02], we use a limit of 10% for $\Delta\varepsilon$ and, in case of violation, compute a reduced step size $h_{K,r}$ such that the condition holds.

[5]For the examples presented in Sec. 4.4, values of $\varepsilon_c^{lim} = 10\%$ and $\varepsilon_n^{lim} = 7.5\%$ were used.

This reduced step size is used for a number n_r of subsequent evaluations, which is determined by requiring that $n_r h_{K,r} = h_K$. Even for the more complex examples considered in this chapter, such step size reductions were only required very rarely. In these cases, however, this technique has proved a valuable support for maintaining stable animations and smooth collision response.

4.3 Asynchronous Collision Handling

In order to be of practical use, it is necessary to account for collision detection and response in the asynchronous framework. We propose a three-stage scheme for this purpose that combines techniques commonly used in synchronous approaches into an efficient asynchronous algorithm.

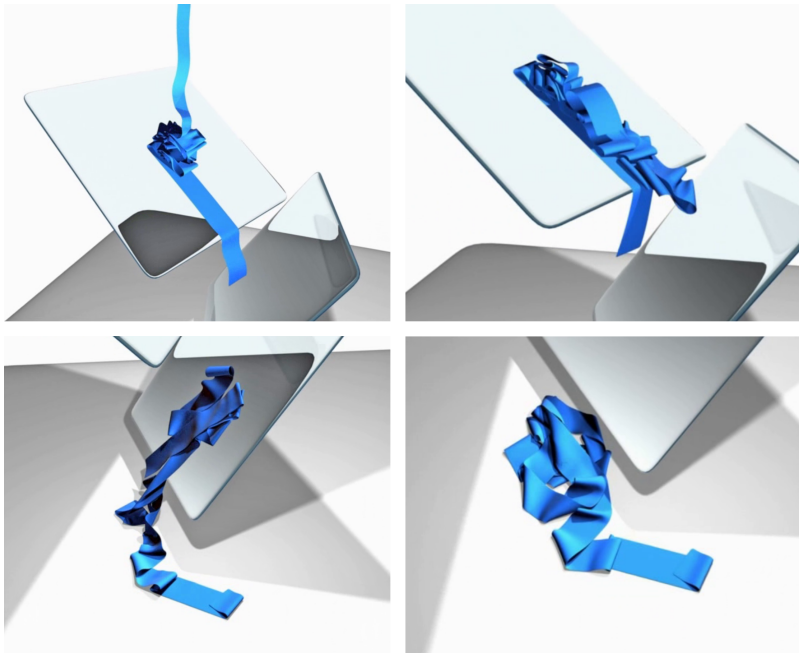


Figure 4.4: Snapshots of an animation sequence exhibiting complex self-collisions with multiple layers of fabric in close proximity.

The method can be outlined as follows. The system is synchronized at regular instants $t_c^i = i \cdot \Delta t_c$, where i is an integer and Δt_c denotes a user-supplied collision step size[6]. The first stage acts on the synchronized state at times t_c^i, detecting triangle pairs that may potentially interfere during the subsequent interval $[t_c^i, t_c^i + \Delta t_c]$. This information is used in the second stage, which couples asynchronous element updates with inter-triangle collision detection and impulse-based collision response. In the third stage, which acts again on the synchronized state, remaining collisions are resolved in an iterative way. The following paragraphs explain each of these stages in detail.

[6]We used $\Delta t_c = 0.01s$ for the animations described in Sec. 4.4.

First Stage The first stage is invoked at each synchronization point t_c^i in order to register the triangle pairs that will interfere in the subsequent interval $[t_c^i, t_c^i + \Delta t_c]$. Since the positions at the end of this interval are unknown, these interferences cannot be determined exactly and have to be estimated instead. To this end, we compute a first-order approximation of $\mathbf{x}(t_c^i + \Delta t_c)$ as

$$\tilde{\mathbf{x}} = \mathbf{x}(t_c^i) + \Delta t_c \mathbf{v}(t_c^i) \,, \tag{4.31}$$

and perform a continuous collision detection pass between $\mathbf{x}(t_c^i)$ and $\tilde{\mathbf{x}}$. This is done in the usual way, i.e., by enlarging the bounding volumes such that they include both the geometry of $\mathbf{x}(t_c^i)$ and $\tilde{\mathbf{x}}$. Having updated the hierarchies of all deformable and rigidly moving objects in this way, recursive overlap tests are performed to obtain a set of potentially interfering triangle pairs. These pairs are subsequently mapped to the cloth objects such that each element is assigned a (possibly empty) list of triangles, which have to be checked for actual collisions in the second stage.

Second Stage The first stage registers potentially interfering triangle pairs, but the actual collision handling is coupled to the asynchronous element updates. After a given element K has been updated at time t_K, it is tested for collisions with all potentially interfering triangles K_i that are registered in its list. To this end, the nodal positions of K_i are synchronized to the current time t_K (using the current nodal velocities of K_i) and the geometric distance between K and K_i is computed as described by Möller [Möl97]. If the distance is smaller than a threshold value, exhaustive tests are performed for all edge-edge and vertex-face pairs of the two triangles. True collisions are then treated as described in Sec. 2.4.2, i.e., by applying impulses that prevent imminent intersections or push too close primitives apart.

Although most of the collisions can be resolved in this way, there can still be intersections at the end of the second stage. Such remaining collisions, which were either not captured by the first stage or not resolved in the second one, are treated in an additional third stage as described next.

Third Stage The task of the third stage is to resolve all collisions that remain after the second stage and we employ an iterative scheme for this purpose. First, a continuous collision detection pass is applied in order to determine potential intersections that occurred during the interval $\left[t_c^i, t_c^i + \Delta t_c\right]$. This is analogous to the first stage, except that the positions $\mathbf{x}(t_c^i + \Delta t_c)$ at the end of the interval are now supplied by the asynchronous stage. Second, continuous intersection tests are performed between the primitives of the detected triangle pairs and collision resolving impulses are applied if necessary. This process is repeated until all intersections are resolved. The integration of rigid impact zones as an additional fail-safe is straightforward, but we found this unnecessary for the examples considered in this chapter.

Continuous Culling The first stage typically captures the vast majority of the actually interfering triangle pairs. It can, however, also lead to a large number

of false positives, i.e., triangle pairs for which no real collisions were detected in the second stage. This effect is especially pronounced for examples with complex self collisions such as the animation shown in Fig. 4.4. Since these false positives slow down the simulation unnecessarily, we aim to reduce their number in an additional culling step.

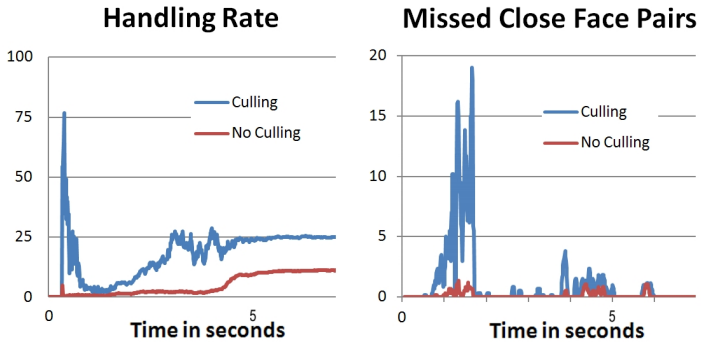


Figure 4.5: Impact of the geometric culling step for the animation shown in Fig. 4.4. The left diagram shows the percentage of actually handled triangle pairs in the asynchronous stage over time. The right diagram plots the number of triangle pairs that were only handled by the third stage.

For each detected triangle pair, a continuous collision test is performed in order to determine whether an actual intersection occurs along the linear trajectory between $\mathbf{x}(t_c^i)$ and $\tilde{\mathbf{x}}$. To this end, we check the corresponding edge-edge and vertex-face pairs for coplanarity during the interval $\left[t_c^i, t_c^i + \Delta t_c\right]$ as described in Chapter 2. If no intersection is found, the triangle pair is removed from the input list for the second stage.

The improvement in the ratio between detected and actually handled triangle pairs can be quite significant: as can be seen in the left diagram of Fig. 4.5, an average culling rate of over 50% was obtained for the ribbon animation shown in Fig. 4.4. The right diagram also indicates that the culling step leads to a slightly higher number of triangle pairs being missed during the second stage[7]. This entails a slight increase in workload for the third stage but we did not observe a negative effect on robustness.

4.4 Results

This section presents numerical results and example animations for the method described in this chapter. We are especially interested in how well surface details are produced and preserved in comparison to existing implicit methods. Although such qualitative aspects are often difficult to quantify and to some extent subjective, the following simple example speaks a sufficiently clear language in this regard.

Example 4.1 consists of a square piece of cloth with 5016 faces, which is pinned at two corners and left swinging for 4 seconds. The material properties were set

[7]Compared to the number of asynchronously handled collisions, the number of missed pairs is still very small.

to $E = 300N/m$ and $G = 150N/m$ for stretching and shearing and a mass density of $\rho = 500g/m^2$ was used. Moreover, we applied deformation limiting with a tolerance of 10% for the synchronous methods, while parameters were set to $\varepsilon_n^{lim} = 7.5\%$ and $\varepsilon_c^{lim} = 10\%$ for our AVI-based scheme. Fig. 4.1 shows qualitative results for this example in form of rendered views, which correspond to the same frame of the animation, but were computed using different integration methods and/or step sizes. The table on the left of Fig. 4.6 lists the corresponding computation times and it can be seen that, when using a large step size of $h = 1/30s$, the semi-implicit (backward) Euler scheme allows simulation at almost real-time rates. The resulting animation is, however, heavily damped and clearly lacks surface detail (see Fig. 4.1, (*a*)) even though no material damping was used.

Integrator	Step size	t_{solve}
Backward Euler	0.0333	5.66
Backward Euler	0.01	17.05
Backward Euler	0.00075	218.4
Forward Euler	0.00020	265.0
Symplectic Euler	0.00025	208.0
Implicit Midpoint	0.001	247.4
AVI-based	0.00037	215.0

Figure 4.6: Comparison of different integration methods on Example 4.1. All values refer to seconds and computation times correspond to the entire animation ($4s$).

The symplectic Euler scheme and our AVI-based method perform significantly better in this regard as can be seen in Fig. 4.1 (*d*) and (*e*), respectively. In order to provide an even comparison, we repeated the example with the implicit solver, this time using a step size of $h = 0.00075s$, which yields roughly the same computation time as for our method. This leads to a decrease in spurious damping such that finer folds and wrinkles can produce (see Fig. 4.1, image (*b*)), but compared to the explicit variants, results still exhibit less detail. We also experimented with a linearized version of the implicit midpoint rule (Fig. 4.1, (*c*)), but found its performance deceiving. Contrary to our purposes, a significant amount of material damping was required for stable simulation, even for a small step size of $h = 0.001s$.

Among the explicit integrators, the forward Euler scheme takes longest to simulate since we could not use as large a step size as for its symplectic counterpart. It is worth noting in this context that the AVI-based method is almost as fast as the symplectic Euler scheme, despite the fact that it has a certain overhead for managing asynchrony. This can be attributed to the fact, that an unstructured mesh with elements of slightly different sizes is used for simulation. The asynchronous method benefits from the freedom to choose elemental step sizes according to local stability requirements, whereas the global step size of the synchronous methods has to satisfy the requirements of the smallest element. For this example, the average time step of the asynchronous scheme was thus slightly higher than for the symplectic Euler method ($0.00037s$ vs. $0.00025s$).

This effect is significantly more pronounced for Example 4.2. The deformable object is represented by a mesh with non-flat rest geometry (see Fig. 4.7) that is

optimized for accurately modeling regions of high curvature with small triangles, whereas large elements are used for flat regions. The animation is quite simple: during 4 seconds of simulation, the mesh is dropped from a small height onto a flat floor, where it comes to rest. The parameters are the same as in the previous example except for a higher bending stiffness, which lets the object recover its shape after the impact. The table on the left hand side of Fig. 4.7 shows the computation times for this example with separate timings for time stepping and collision handling. The backward Euler scheme is again the fastest, but the AVI-based method performs significantly better than the other explicit schemes and the implicit midpoint rule.

Integrator	Step size	t_{solve}	t_{coll}
Backward Euler	0.005	7.7	4.3
Backward Euler	0.001	16.1	9.7
Forward Euler	$1{\cdot}10^{-5}$	571.5	213.3
Symplectic Euler	$5{\cdot}10^{-5}$	109.5	62.1
Implicit Midpoint	0.0001	323.9	52.0
AVI-based	0.0002	47.2	21.1

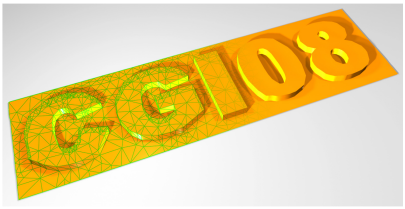


Figure 4.7: Comparison of different integration methods on Example 4.2, which consists of an unstructured mesh with 2304 elements of considerably different size (the ratio between the areas of the largest and smallest element is 160:1). All values refer to seconds and computation times correspond to the entire animation (4*s*).

While the first two examples are of a rather technical nature, we also investigated the capability of our method to generate more complex animations, two of which are described in the following. In Example 4.3, a woman wearing a long dress (comprised of 8891 faces) walks through an obstacle course, along which she climbs a number of stairs and takes several additional hurdles (see Fig. 4.8).

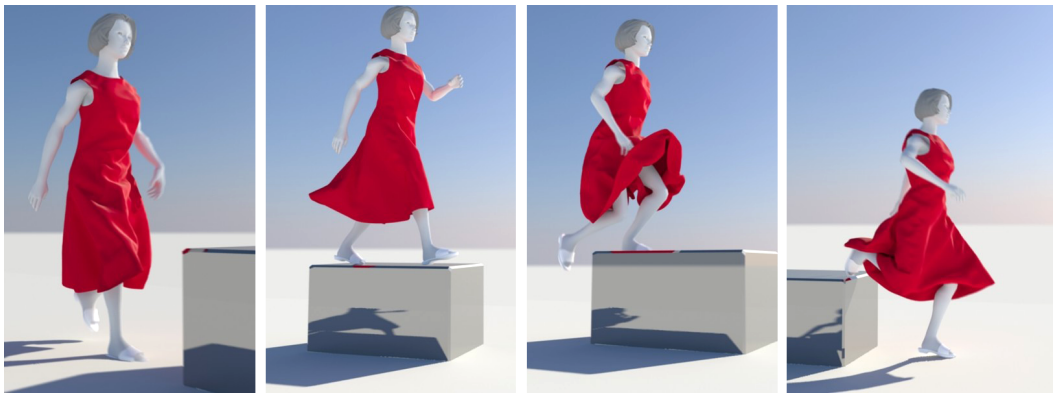


Figure 4.8: Snapshots from an obstacle-walk animation (Example 4.3).

Cloth simulation with animated characters is a common application scenario and this example puts special emphasis on cloth-object collisions in combination with fast rigid body motion. Our method deals with these challenges in a smooth and robust way and generates vivid motion of cloth with rich surface details. For comparison, we also ran this example with the semi-implicit Euler method for time integration. As to the computational performance, the latter was roughly

30% faster than our method when using a step size of $h = 0.001s$. The additional computation time is, however, compensated for by less damped motion and more detailed folds and wrinkles, as can be seen in the comparison shown in Fig. 4.9.

While external collisions were predominant in the previous animation, the last example puts special emphasis on self-collisions. A long ribbon consisting of 8000 faces is positioned in vertical orientation above two inclined planes (see Fig. 4.4). During 5 seconds of animation, the ribbon falls onto the planes and slides towards the floor, forming complex self collisions along its way. Even for the most challenging frames with multiple layers of fabric pressed tightly together, all collisions are resolved gracefully and without taking recourse to a fail-safe such as rigid impact zones. This is due to the comparatively small elemental step size ($h_{avg} = 0.00025s$) and the corresponding high rate at which collisions are handled. For this example, a single 25Hz frame took an average of 185 seconds to compute and, due to the large collision step size of $\Delta t_c = 0.01s$, more than 90% of this time was spent in the asynchronous update loop.

Figure 4.9: Comparison between implicit Euler (*left images*) and our AVI-based method (*right images*) on two frames from Example 4.3.

4.5 Conclusion

This chapter described a method that leverages asynchronous time stepping for efficient simulation of low-damped cloth. While approaches based on semi-implicit time integration are plagued by numerical dissipation, our method does not suppress fine folds and wrinkles, thanks to the explicit nature of the underlying integrator. In order to increase the efficiency and robustness of the latter, we implemented a strategy that exploits the local time stepping for monitoring and limiting deformations at run time. Drawing on techniques from the synchronous setting, we have further proposed an efficient method to incorporate collision handling into the asynchronous framework.

Our method aims at high-quality animations and we have demonstrated its capabilities in this regard on two complex examples. Compared to a standard approach based on the semi-implicit Euler scheme, the increase in computation times seems justified by the higher degree of detail observable in the results.

A recent development that should be mentioned in this context is the work of Harmon et al. [HVS+09], who presented a rigorous treatment of contact mechanics for deformable models, using asynchrony in order to treat collisions in true order of occurrence. In addition to asynchronous variational integration, the enabling technology of this approach are kinetic data structures and nested contact potentials for collision detection and response, respectively. This allows simulations with remarkable accuracy and robustness, but computation times are (currently) beyond what is affordable for practical applications. Our method does not strive to attain this extreme level of accuracy and rather focuses on a careful selection of approximations that enable efficient computations. However, in contrast to the standard approach based on semi-implicit time integration, this is achieved with less compromise in visual quality.

As a possible direction for future work, spatially adaptive discretizations seem a promising perspective to increase the computational efficiency of our method. In this context, the efficiency of AVIs for discretizations with differently sized elements is particularly attractive. Additionally, their explicit nature make dynamic local adaptations a lightweight process, whereas the Jacobian matrices of implicit schemes are rather reluctant to these changes.

Chapter 5

Parallel Collision Handling

A robust handling of collisions and contact is vital for creating complex animations of deformable surfaces. State-of-the-art collision processing methods can deliver compelling results even for challenging scenarios with multiple, tightly-packed layers of cloth (see Fig. 5.1). However, this robustness comes at the price of intensive computations, which can easily take up the largest part of the total simulation time. This chapter presents a collision handling method that leverages the processing power of modern parallel computers in order to accelerate these computations.

5.1 Introduction

As parallel processing power becomes widely available, the demand for efficient algorithms that can exploit these resources also propagates into the field of computer animation. This is especially true for physically-based simulations of deformable surfaces and solids, for which time integration and collision handling are particularly time-consuming. In order to accelerate such computation-intensive methods on parallel platforms, it is crucial to distribute work evenly among processing units such as to utilize available resources as much as possible. For this purpose, the problem has to be decomposed into a number of subproblems.

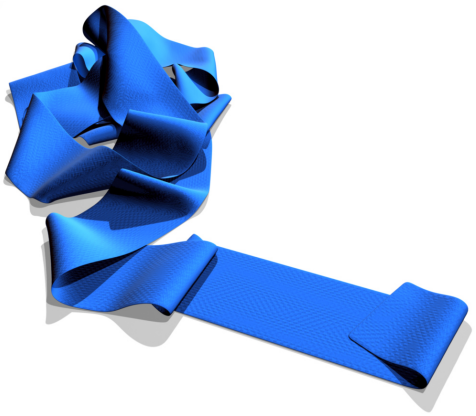


Figure 5.1: Complex multi-layer self collisions.

Implicit time integration is a regularly structured problem with predictable interaction and memory access patterns[1], which lends itself well to static problem decomposition. Dividing the input mesh into a number of partitions induces a decomposition of the arising linear system, which can be used as the basis for a parallel solution method (see, e.g., [KB04]). Since the structure of the system stays

[1] see [GRV95] for a taxonomy of problems in the context of parallel computing

constant over time, the decomposition has to be computed only once.

By contrast, collision handling for deformable surfaces is a highly *irregular* problem since the number and locations of collisions change rapidly over time and cannot be predicted in advance. A static problem decomposition is bound to produce a high degree of processor idling and, consequently, results in poor efficiency. In order to obtain good parallel performance for such difficult problems, a dynamic problem decomposition is mandatory.

Previous Work Although several approaches have been presented for accelerating cloth simulation on parallel computers [RRZ00, LGPT01, ZFV02, KB04, ZFV04, GRR+05], few propose a specific treatment of collisions. In particular, no previous cloth simulation method addressed parallel collision handling on distributed-memory architectures.

A notable method for shared-memory parallel computers is due to Romero et al. [RRZ00], who describe a data-parallel approach based on bounding volume hierarchies and separation lists[2] [LC98]. The basic idea is to perform a sequential collision detection pass to obtain a separation list, whose entries are then distributed evenly among the processing units. Good parallel efficiency can be expected if the separation list has to be recomputed only infrequently, i.e., for quasi-static scenes. However, in order to maintain well-balanced workloads for practical animations with rapidly changing collisions, frequent rebuilds are required. Consequently, the sequential hierarchy detection is likely to become the limiting factor for parallel efficiency.

Parallel collision detection has also been studied in the context of engineering applications such as impact simulation. Methods from this field typically rely on partitioning input data, which can be accomplished via spatial subdivision [PAH+98, BASH00] or graph partitioning [Kar03]. At any rate, the decompositions have to be adapted or recomputed regularly in order to maintain well-balanced workloads. Although good scalability was reported for all these approaches, they were designed for solid simulations in which the potentially colliding surface polygons constitute only a small fraction of the total number of elements. It is unclear to what extent efficiency can be maintained in the case of cloth simulation, where every pair of elements can potentially collide.

Overview and Contributions Instead of relying on static data partitioning, we propose to treat the irregular nature of collision handling with a task-parallel approach based on fully dynamic problem decomposition. Using bounding volume hierarchies for collision detection and stopping impulses for collision response, our parallel algorithm extends widely used sequential techniques to the parallel setting. The basis of this approach is formed by a reformulation of recursive collision detection as a depth-first tree traversal, which is amenable to parallel execution. Parallelism is generated dynamically by decomposing bounding volume tests into subtasks, which can be sent to remote processing units for execution. This mechanism is steered by a dynamic load balancing scheme, which uses a distributed task pool model in order to achieve a high degree of resource utilization.

[2] although the authors did not use this terminology

We implement this strategy using multithreaded programming and analyze the parallel efficiency of the resulting approach on a number of examples.

The method described in this chapter was developed for distributed memory architectures and was presented in [TB06a] and [TB07]. Although we restrict considerations to this setting here, our approach is readily translated to shared-memory architectures such as multi-core platforms [TPB07, TPB08].

The remainder of this chapter is organized as follows. Building on the sequential framework described in Chapter 2, Sec. 5.2 describes how the potential parallelism in recursive collision detection is exposed and exploited. Corresponding strategies for problem decomposition and dynamic load balancing are presented in Sec. 5.3. We analyze the performance of our approach in Sec. 5.4 and conclude with a summary in Sec. 5.5.

5.2 Exposing Parallelism in Collision Handling

Our method for parallel collision handling builds on the sequential basis as described in Chapter 2: it uses k-DOP hierarchies [KHM+98, MKE03] for collision detection and an impulse-based collision response [Pro97, BFA02]. This approach can be considered state-of-the-art, but it does not directly lend itself to a parallel implementation. This section describes how to reformulate the recursive collision detection in order to expose its potential for parallelization.

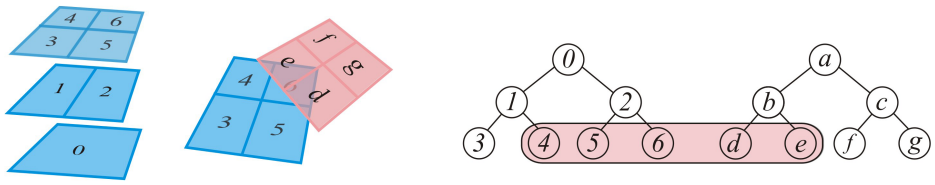


Figure 5.2: Schematic view of the bounding volume hierarchies for two interfering rectangles. *Left*: hierarchy structure and object arrangement. *Right*: overlapping leaf nodes marked in the tree structure.

An observation fundamental to our strategy is that recursive testing of two bounding volume hierarchies can be considered as a tree traversal [LC98]. Fig. 5.2 shows a simple example in order to illustrate this view. The root of the recursion tree T, which is shown in Fig. 5.3, corresponds to the overlap test between the roots of the two bounding volume hierarchies H_1 and H_2. Every other node $N_{ab} \in T$ corresponds to an overlap test of two DOPs $D_a \in H_1$ and $D_b \in H_2$. The children of N_{ab} are defined as the set of all pairwise tests between the children of D_a and D_b. Note that the recursion tree does not have to be binary, but at any rate, it is only a conceptual construct that is never built in practice.

We implement the recursion tree traversal as a depth-first search, which is particularly well suited for parallel implementation [RK87]. The process of depth-first traversal starts at the root of the test tree. If an overlap is detected, one of the

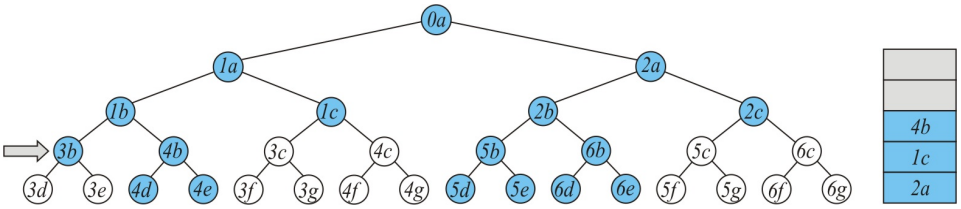


Figure 5.3: Test tree for the colliding objects shown in Fig. 5.2.

nodes is expanded into its n child nodes, resulting in n bounding volume tests[3]. Instead of invoking a recursive function call on each newly generated test, the depth-first traversal continues with one, e.g., the left-most test while the remaining nodes are pushed onto a *stack*. This process continues downwards in the tree until a leaf is reached. Upward traversal begins by fetching a node from the top of the stack and starting a new depth-first traversal on the corresponding subtree. The process finishes when the stack is empty, which implies that all nodes in the testing tree have been visited. Note that this is an exhaustive search, which is exactly equivalent to recursive testing. However, the central difference is that the stack structure can be exploited for creating parallelism: each stack entry corresponds to an untried branch of the recursion tree and entries are mutually independent. In particular, changing the order or processing entries elsewhere does not affect the exhaustiveness, i.e., the correctness of the traversal. Practically, this means that we can freely remove entries from the stack, assign them to *tasks*, and send these tasks to remote processing units for execution.

The optimal strategy of task creation and distribution depends on the platform under consideration and details for the distributed-memory case will be given in the next sections. At any rate, good parallel efficiency requires that the number of synchronization points be kept at an absolute minimum. We therefore integrate hierarchy traversal, decomposition into primitives, geometric distance tests and impulse generation into a single processing stream. This strategy minimizes synchronization overhead and maximizes the ratio of computation to communication.

5.3 Problem Decomposition

Clusters of interconnected computers are typically programmed as distributed-memory machines. This setting invites an SPMD (single program, multiple data) programming style, in which all processing units execute the same operations but on different data. The Message Passing Interface (MPI) provides extensive support for this programming model through comprehensive communication and synchronization functionality. Previous work by Keckeisen and Blochinger [KB04] introduced an efficient method for cloth simulation using parallel implicit time integration. This section describes how parallel collision handling can be integrated into this framework.

[3]It is also possible to test the children of both nodes directly against each other, which would results in $2n$ tests. However, expanding only one node immediately (e.g., the one with the larger bounding volume) and thus creating only n tests has proved more efficient in practice. See [MKE03] for details.

5.3.1 Static Problem Decomposition

The parallel time integration scheme of [KB04] is based on a static decomposition of the input mesh into a number of disjoint partitions (see Fig. 5.4 for an example). This decomposition is computed with the parallel multilevel graph partitioning functionality provided by the ParMetis library [KK96b]. On the basis of this partitioning, each processing unit is assigned a certain subset of the vertices and faces of the input mesh. In order to achieve a maximum degree of locality, it is reasonable to account for this decomposition when building the bounding volume hierarchy. To this end, we construct a bounding volume hierarchy for each partition and combine the root nodes of the different processing units into a global hierarchy. On this basis, we formulate collision detection for a deformable surface mesh as a number of *top-level tasks*: each partition is tested against every rigid object in the scene, against all other partitions of the same deformable surface and against itself.

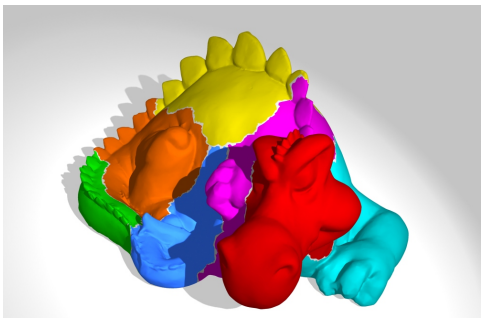


Figure 5.4: A *'cloth dragon'* decomposed into 12 partitions [TWW+07].

A naive approach would distribute top-level tasks evenly among the processors, but this is insufficient for achieving high processor utilization in dynamic scenes. In Sec. 5.4 we present examples where such uneven processor loads can be observed. In order to obtain an efficient implementation with well-balanced workloads, we propose a task-parallel approach which dynamically decomposes top-level tasks into smaller subtasks. This method is described in the following section.

5.3.2 Dynamic Problem Decomposition

Our method for dynamic problem decomposition is based on the depth-first search formulation of bounding volume hierarchy testing described in Sec. 5.2. Each processing unit is initially assigned one or several top-level tasks. During the depth-first traversal of these top-level tasks, untried test nodes are pushed onto a local stack. There are two options for processing entries from the stack. A test can be removed from the top of the stack and executed by the task that performs the current traversal. This case corresponds to the sequential version of the depth-first search. The other option is to remove one or several tests from the bottom of the stack and to assign them to a newly generated subtask. This subtask can either be executed locally or be sent to a remote processing unit. Either way, some tasks will eventually encounter interfering triangle pairs and return corresponding stopping impulses. All these partial responses are combined at the end of the collision handling phase by one all-to-all broadcast operation and applied to the positions vector.

A central aspect in the creation of subtasks is an adequate control of granularity. Since tasks are likely to be transferred to remote processors, they should at least represent a certain minimum amount of work in order justify the incurred

communication overhead. However, it has to be ensured as well that there are always enough tasks available for incoming transfer requests. For these reasons, we pursue an adaptive decomposition strategy that is steered by the dynamic load balancing process described in the following section.

5.3.3 Dynamic Load Balancing

The dynamic load balancing process is responsible for triggering and coordinating task creation and task transfer operations in order to prevent processors from running idle. Load balancing can either employ a central controller or be organized in a distributed manner. A central controller can establish a more accurate view of the current processor loads, but it runs the risk of becoming the sequential bottleneck of a parallel algorithm. In order to achieve high scalability, we employ a fully distributed scheme. In particular, each processor maintains a local *task pool*, where tasks are initially placed upon creation. Tasks can then be instantiated and executed locally or be transferred to a remote task pool for load balancing purposes. As described subsequently, task creation and transfer operations are initiated autonomously by the processing units.

Task Creation Tasks are generated dynamically on the basis of the decomposition scheme described in the previous section. In order to achieve a high degree of efficiency, an active processing unit must be able to satisfy task requests from other processors immediately. But additionally, it has to be ensured that tasks represent a sufficient amount of work. This, however, means that the stack must contain a minimum number of entries before tasks can be created. In order to meet both requirements, we generate tasks in a proactive fashion, i.e., independently of incoming tasks requests. Tasks are generated whenever the size of the stack exceeds a threshold value τ. Task generation generally creates overhead, even when the new task is subsequently executed on the same processor. We therefore increase τ linearly with the current size of the task pool σ as $\tau = a\sigma + b$, where a and b are scalar parameters. A new task is assigned $\tau/2$ tests, which are taken from the bottom of the stack. Such tests have a higher potential of representing a large amount of work, since they originate closer to the root of the current testing tree. With this strategy, we can rapidly create tasks of minimal granularity (determined by b) and at the same time ensure that subsequently generated tasks represent an increased amount of work such as to compensate creation overhead. The parameter values a and b largely depend on the parallel architecture used and should be determined experimentally. We found that choosing $a = 2$ and $b = 8$ delivered consistently good results for all examples considered.

Task Transfer We employ a receiver initiated scheme for transferring tasks between processing units. When a processor runs idle and its local task pool is empty, it requests work from remote pools. A target node to be sent the request is chosen on the basis of a round robin scheme. If available, the target node transfers a task from its pool to the local pool. Otherwise the request is rejected and the process is repeated for another target node.

Task Representation The dynamic load balancing scheme transfers tasks between processing units at run time. In distributed memory architectures, task transfers require explicit communication operations, which can constitute a significant part of the overall parallel overhead. A compact description is therefore crucial in order to minimize communication overhead. In our case, the costs for transferring a task are largely determined by the representation of the associated bounding volume hierarchy tests.

A prerequisite for dynamic load balancing is that task execution is independent of location. Endowing each task with the complete set of data required for its execution is very wasteful in terms of communication costs. Instead, we choose to replicate the required data, i.e., the bounding volume hierarchies of all objects in the scene on every node locally. This provides an identical context for task execution on every node and enables us to represent tasks in a compact way using only 4 indeces (H_1, H_2, D_1, D_2): H_1 and H_2 identify the hierarchies while $D_1 \in H_1$ and $D_2 \in H_2$ refer to corresponding bounding volume nodes, which define the root for the test.

The size of the bounding volume hierarchies is comparatively small for usual scenes, even if high-resolution models are used. The additional storage requirements of this approach are therefore negligible in practice. The run time overhead remains small as well since updating the hierarchies requires only an all-to-all broadcast operation in order to provide all processors with the complete position vectors of all objects. The structure of the hierarchies is kept fixed during the whole simulation such that no additional costs occur due to rebuilding. It has to be pointed out, that the latter is not an optimization made in order to improve the efficiency of the parallel algorithm. Rather than that, we did not observe a consistent performance improvement for the sequential version when regularly rebuilding the hierarchies.

5.4 Numerical Experiments and Results

This section provides a performance analysis of the parallel collision handling scheme described in this chapter. We will first briefly comment on implementational aspects and describe the test system as well as the examples used for the performance measurements.

5.4.1 Implementation and Example Setup

Implementation Implementing the parallel collision handling scheme on distributed-memory architectures requires a framework for distributed multithreading. We use the parallel system platform DOTS due to Blochinger et al. [BKLW99], which provides extensive support for the multithreaded parallel programming model. In particular, DOTS includes functionality for termination detection, thread transfer between nodes and event-handlers for implementing load balancing strategies. Implementational details with respect to DOTS can be found in [TB07].

Testing and Measuring The performance measurements were carried out on a Linux-based cluster with 12 computation nodes, equipped with Intel Xeon CPUs (2.667 GHz) and 2 GB of main memory. The nodes are connected by a Myrinet-2000 network.

The example animations consist of 1000 simulation steps and a step size of $0.001s$ was used. The measurements are computed as the arithmetic mean of the wall-clock times of three individual runs. Timings for the single processor case are based on a sequential version of the simulator. In particular, only sequential data structures and sequential arithmetic operations are used for this configuration and no dynamic problem decomposition was performed during the collision handling phase.

Example Setup We evaluate the performance of our approach using two examples and in order to demonstrate the robustness of our method, we focus on problems with a high degree of irregularity. Additionally, we aim at accelerating commonly used scenarios and therefore restrict considerations to input data of moderate size.

For the first example (*Example 5.1*), a disc-shaped table cloth with 14.033 vertices and 27.532 faces is draped over a sphere with roughly on third of the cloth's diameter (see Fig. 4.8). Initially, collisions occur only in a locally bounded region near the center of the cloth. As the simulation proceeds, the distribution of the collisions becomes more even. The folds formed in the last part of the animation lead to massive intra- and inter-partition self collisions.

Figure 5.5: Representative rendered frame (*left*) with corresponding mesh partitioning (*right*) for Example 5.1.

In the second example (*Example 5.2*) a square piece of cloth with 14.641 vertices and 28.800 faces is draped over a thin wave-shaped bar, which is posed at some distance to a rigid floor (see Fig. 5.6). Again, collisions are locally bounded in the first part of the simulation. Due to the shape of the bar, complicated folding patterns are formed as the cloth falls further downwards. When it reaches the floor, rigid collision occur throughout large parts of the mesh and the setting becomes more regular. Although rigid collisions are initially predominant, also self collisions occur between the detailed folds as the cloth slides towards the troughs of the bar.

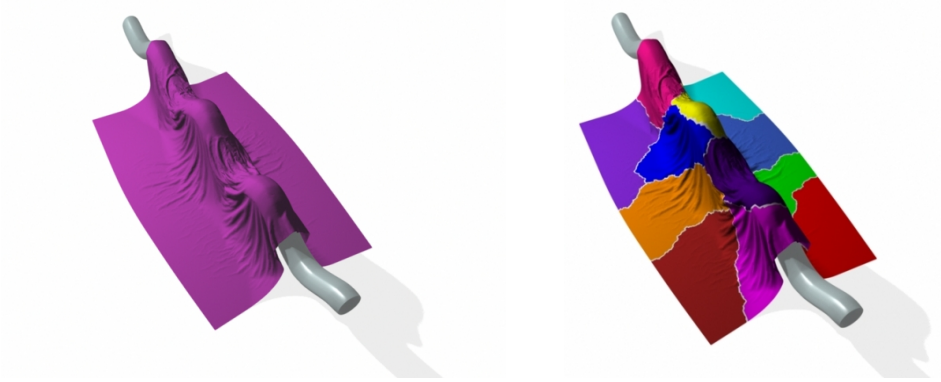


Figure 5.6: Representative rendered frame (*left*) with corresponding mesh partitioning (*right*) for Example 5.2.

5.4.2 Results

The diagrams in Figs. 5.7 and 5.8 show the results of the performance measurements for the two examples. The costs for time integration of the physics (referred to as *physical modeling*) and preprocessing are included for comparison. Despite the high degree of irregularity of the examples, the overall computation time as well as the individual run times for both time integration and collision handling are substantially reduced by parallel execution. In particular, very good parallel efficiency is achieved for the collision handling stage.

In both examples, collision handling takes longer than time integration. While the latter requires a comparable amount of time in both cases, the higher number of self collisions in Example 5.1 also entails higher costs for collision handling in comparison to Example 5.2. Regardless of the number of processors, we observe a constant amount of time spent on preprocessing, which is largely due to mesh partitioning and bounding volume hierarchy setup. However, for typical production runs with several hundreds of frames, this overhead becomes negligible.

In order to assess the influence of our dynamic problem decomposition and load balancing scheme, we compare two program runs with 12 processors for each of the test scenes. The diagrams shown in Fig. 5.9 illustrate the average CPU utilization of each processor for the collision handling stage during the course of the animation. The diagrams on the left and right side show data for the runs in which problem decomposition and load balancing were disabled and enabled, respectively. Additionally, the total time spent on collision handling is given for each frame in the left diagrams, while corresponding acceleration factors are given in the right diagrams. Note that the computational costs of collision handling increase by one to two orders of magnitude when complex collisions occur.

The diagrams reveal that the combination of dynamic problem decomposition and load balancing leads to a considerable improvement in parallel efficiency. This increase is particularly clear for Example 5.1, which exhibits a higher degree of unevenly distributed (self-)collisions. No interactions occur during the first frames and we observe slight slowdowns, which reflect the overhead introduced by the dynamic approach. This overhead is, however, marginal in comparison to the improvement in overall efficiency.

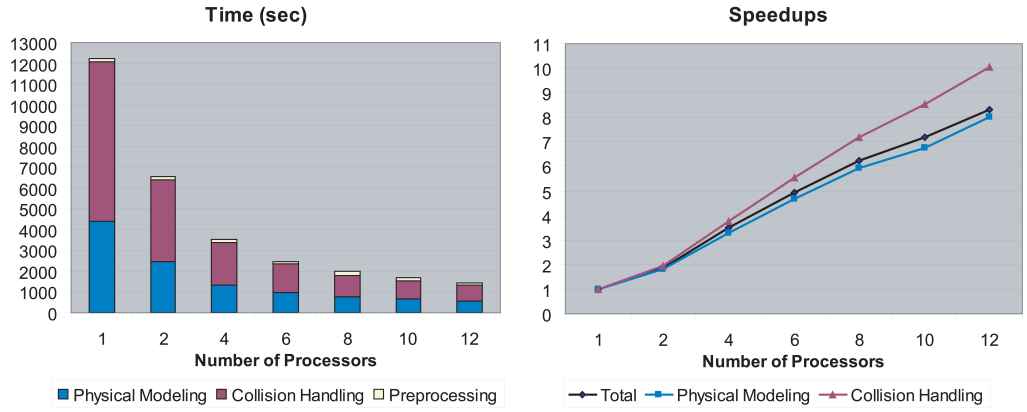


Figure 5.7: Results of performance measurements for Example 5.1.

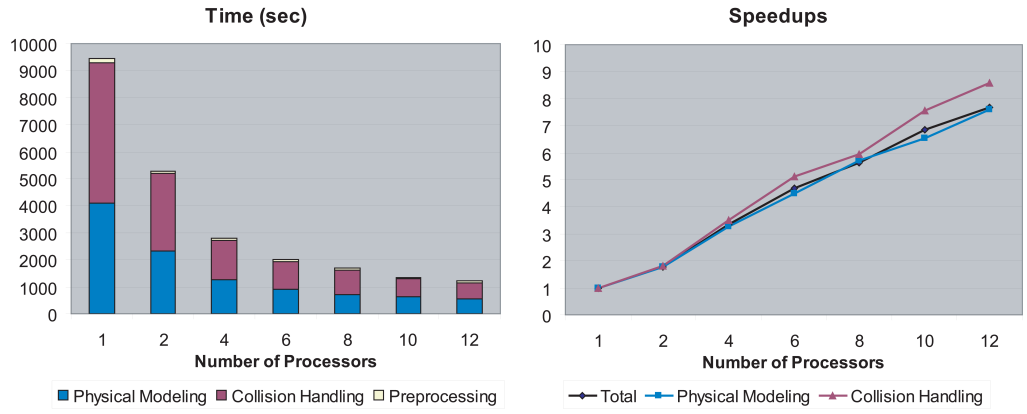
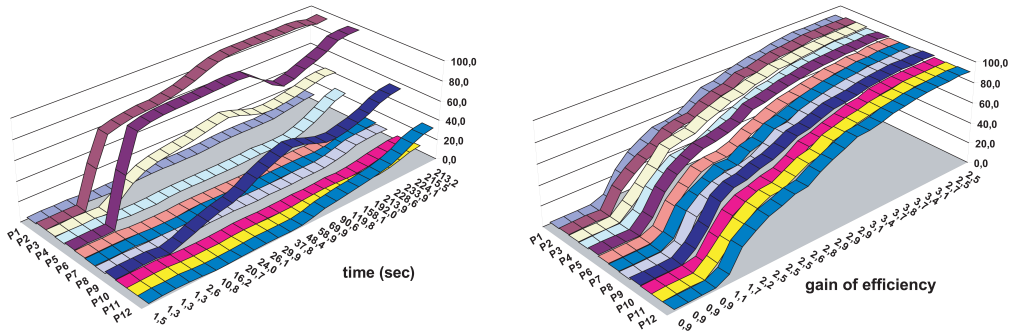
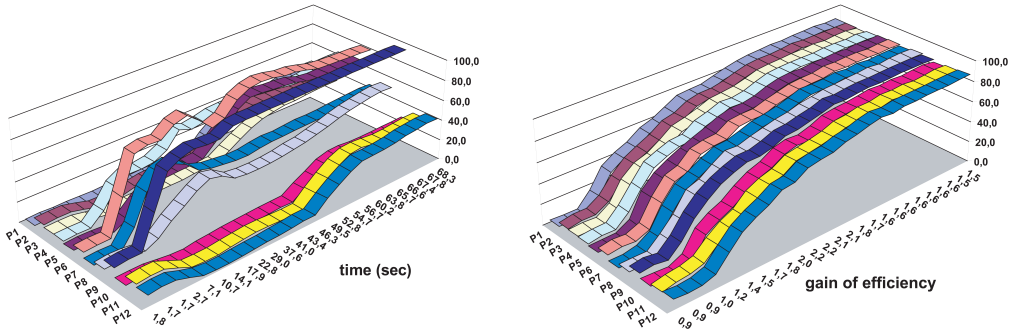


Figure 5.8: Results of performance measurements for Example 5.2.

To summarize, the results presented in this section indicate a high parallel efficiency for our method. Its robust performance can be attributed to the fact that problem decomposition and load balancing are tightly interrelated, thus achieving self-adapting parallelism. For regular examples where collisions are evenly distributed among the processors, the overall amount of dynamic parallelism is limited. By contrast, if processors run idle due to an uneven distribution of collisions, additional parallelism is generated and balanced across the processors. Our task-parallel collision handling approach integrates seamlessly into the SPMD framework and, in particular, has no negative effect on the performance of the data-parallel time integration scheme. This combination can therefore be considered a successful symbiosis of considerably different types of parallelism within a single application.



(a) Example 5.1



(b) Example 5.2

Figure 5.9: Effect of dynamic problem decomposition and load balancing on the parallel efficiency of the collision handling phase. For the diagrams in the left/right column, dynamic load balancing was disabled/enabled.

5.5 Conclusion

Summary This chapter presented a parallel approach for handling collisions between deformable surfaces. Our method extends state-of-the-art sequential techniques to the parallel setting: collision detection based on bounding volume hierarchies and impulse-based collision response.

Collision handling is an irregular problem in which interactions cannot be predicted in advance. Since a parallel approach based on static problem decomposition is insufficient in such settings, we pursued a dynamic strategy for this purpose. In order to expose its potential parallelism, we have cast the recursive collision detection on bounding volume hierarchies as a depth-first tree traversal. This view allows us to generate parallelism dynamically by assigning subtrees of the recursion tree to individual tasks. The latter are then distributed among the available processing units in order to accelerate computations.

In order to implement this strategy, we proposed a task-parallel approach based on multithreaded programming and a distributed task pool model for dynamic load balancing. By making their execution independent of location, tasks can be transferred dynamically between pools on different nodes in order to distribute work evenly among processing units. The benefits of this dynamic load balancing scheme were assessed experimentally and up to threefold accelerations

over a naive static decomposition were obtained. As a result, a high overall parallel performance was achieved.

Subsequent Work and Future Directions Parallel collision detection and response is the subject of ongoing research and several new contributions have emerged from this field. An extension of our method to shared-memory architectures showed good parallel efficiency for several recent multi-core platforms [TPB07, TPS08]. A development aimed at the same architecture is due to Tang et al. [TMT09], who translated sequential acceleration techniques such as advanced triangle culling or separation lists to the parallel setting. Apart from bounding volume hierarchies, also space partitioning methods have potential for efficient parallel implementations [LK02]. Sparse grids, which only store data for occupied cells, are mandatory in this context in order to contain storage requirements. Efficient random access to cells requires hash functions, but previous methods suffered from index collisions. However, the collision-free spatial hashing due to Lefebvre and Hoppe [LH06] might turn sparse voxel grids into a competitive alternative for parallel collision detection. Another line of research is to leverage the capacities of computer clusters in order to explore problem sizes that cannot be treated by single workstations [SSIF08]. Especially for the latter category, it would be interesting to investigate hierarchical distributed-memory systems, in which each node consists of several multi-core processors. This scenario is of particular practical interest for low-budget high performance computing, since a system assembled from a few interconnected multi-core workstations can already concentrate a significant amount of computation power.

Chapter 6

Magnetic Interaction for Rigid Body Simulation

Deformable surfaces and solids constitute only a small part of virtual environments and rigid bodies typically dominate the scene in video games and film. Given the fact that many rigid bodies are made of metal, it is a notable lack that *magnetic interaction* has been largely overlooked in this context. This chapter presents an accurate and efficient method for modeling such magnetic interactions in rigid body simulations.

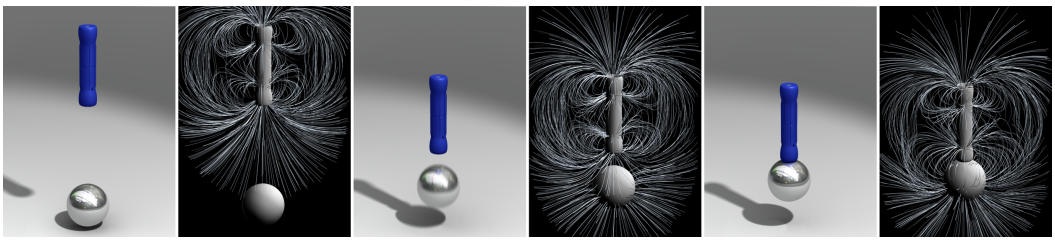


Figure 6.1: Interleaved renderings of animation frames and magnetic field. A toy magnet, carrying small permanent magnets in its ends, attracts a metallic sphere.

6.1 Introduction

Simulating rigid bodies and their dynamic interaction through collisions and contact began to attract the interest of computer graphics researchers more than twenty years ago [Hah88, MW88, Bar89a]. Since these initial developments, substantial progress has been made regarding frictional contact [Bar94, ST96, GBF03], control of animations [Coh92, PSE+00, TJ08] and coupling with other physically-based simulations [OZH00, CMT04, SSIF07], to name just a few. To the best of our knowledge, no attempt to model magnetic interaction between rigid bodies has been reported in computer graphics so far.

Accounting for magnetic effects opens a range of possibilities for creating stunning animations that cannot easily be achieved with artificial forces or inverse kinematics. Similarly, video games can benefit from magnetic interactions at real-time frame rates but also the visualization of magnetic field lines in physics classes

could be complemented or replaced by computer graphics techniques (see Fig. 6.1).

Computing magnetic fields and forces on magnetized objects is a problem frequently encountered in applications from electrical engineering. A standard approach is to first solve the governing, i.e., Maxwell's equations using a conventional finite element method in order to obtain the magnetic field throughout the region of interest [MSP88]. The magnetic forces and torques acting on objects in this field can then be computed with one of many approaches, including the virtual work method, Maxwell's stress tensor method or equivalent source methods. We refer to de Meiros et al. [dMRM98] and Delfino [Del01] for an overview of these methods and their numerical implications. Solving Maxwell's equations with finite elements is certainly accurate, but also too expensive for interactive applications.

Only few works from computer graphics have addressed magnetism or magnetic interaction. Klein and Ertl [KE04] simulate ellipsoidal particles in order to visualize magnetic field lines. The interaction between particles and with the external field is formulated as a simplified energy minimization problem, which is solved with conjugate gradients. Explicit expressions for force and torque between particles are not derived. Remotely related is also the work of Kim and Lin [KL04], who model lightning and electrical arcs using a physically-based approach. A simplified version of the Helmholtz equation is used for modeling the propagation of electromagnetic waves.

6.1.1 Overview and Contributions

This chapter presents a model for discrete magnetic interaction that is simple and fast while reproducing all macroscopic magnetic effects in a faithful way. Instead of relying on costly discretization methods, we take an approach which is discrete from the ground up. Our method builds on the interaction between pairs of magnetic dipoles, for which closed-form expressions can be obtained. The central properties of this method are summarized below.

Symmetric Magnetic Dipole Approach We pose magnetic force and torque computations as discrete *magnetic dipole* interactions and provide explicit formulas for all quantities. The basis of our method is to model magnetic objects as aggregates of magnetic dipole cells. This approach is consistent since the dipole approximations for field, forces and torques converge to their continuous counterpart with increasing number of cells. The method presented in this chapter is similar to the equivalent magnetic dipole approach described, e.g., in [DMMR01]. However, our approach is *symmetric* since we compute both forces and fields from dipole contributions. This avoids the computation of magnetic fields via FEM solvers and automatically ensures conservation of linear and angular momenta. Conserving such physical invariants is important since it ensures a correct dynamic behavior even if magnetic properties are only coarsely approximated.

Adaptive Refinement The accuracy of an object's dipole field approximation depends on the cell density. However, the magnetic field decays rapidly away from

its source and indeed resembles a dipole field at large enough distances. When computing interactions between pairs of magnets, we can therefore save computation time by adapting cell resolution as a function of inter-object distance. For this purpose, we propose an adaptive sampling scheme based on a hierarchical multi-resolution object partitioning. Computational efficiency is greatly increased in this way and, as a result, dozens of magnetic objects can be simulated at interactive rates.

Magnetic Materials There exists a variety of magnetic materials with very different properties. We provide physical descriptions and implementations for hard ferromagnets or permanent magnets, soft ferromagnets, paramagnets, diamagnets and even superconductors.

Validation The accuracy of the proposed method is assessed on a problem with known analytical solution and simulation results are compared to real-world experiments. Additionally, we investigate the influence of the sampling density on approximation quality by comparison with exact results. Finally, we provide example simulations that demonstrate the usefulness of our method for practical computer animation.

The remainder of this chapter is organized as follows. The next section describes magnetic dipole interactions and derives closed-form expressions for fields, forces and torques. Magnetic materials and their implementations are also described within this context. Sec. 6.3 gives an overview of implementational aspects and in particular describes the adaptive cell sampling scheme. Sec. 6.4 presents an extensive set of example animations and provides quantitative performance data in terms of computation times. The chapter closes with a conclusive summary in Sec. 6.5.

6.2 Physical Modeling

In order to simulate magnetic interactions between aggregates of dipole cells, expressions for magnetic fields, forces and torques are required. We start with a brief overview of the relevant parts of magnetostatics. Most of these concepts are well-known and we refer to the standard textbooks by Jackson [Jac99] and Landau et al. [LLP84] for detailed explanations.

6.2.1 Magnetostatics

We will assume that magnetic objects are uncharged and that no time-dependent electric currents are present. This reduces the interaction of magnetized objects to a magnetostatic problem. The relevant part of Maxwell's equations[1] reads

$$\nabla \cdot \mathbf{B} = 0, \qquad \nabla \times \mathbf{H} = \mathbf{J}\,, \tag{6.1}$$

$$\mathbf{H} = \frac{1}{\mu_0}\mathbf{B} - \mathbf{M}\,, \tag{6.2}$$

[1] in SI units

where the last equation is a constitutive relation linking **B** and **H**. In these equations, **B** denotes the magnetic flux density or magnetic induction, **H** the magnetic field and **J** the current density. Furthermore, μ_0 denotes the permeability of free space and **M** the magnetization or density of the magnetic moment. For a given current distribution **J** we define the associated magnetic moment **m** as

$$\mathbf{m} = \frac{1}{2}\int_V (\mathbf{x} - \mathbf{r}_O) \times \mathbf{J}(\mathbf{x} - \mathbf{r}_O)\, dx\,, \tag{6.3}$$

where $\mathbf{r}_O$ is an arbitrary point of reference inside the volume V occupied by **J** (see also Fig. 6.3). The magnetic induction **B** can be obtained by a series expansion of the magnetic vector potential for distances **r** large compared to the spatial extent of the magnetized volume V [Jac99]. The leading term of the magnetic induction for a magnetic moment **m** located at position $\mathbf{r}_O$ reads

$$\mathbf{B}(\mathbf{r}) = \frac{\mu_0}{4\pi}\left[\frac{3\mathbf{n}(\mathbf{n}\cdot\mathbf{m}) - \mathbf{m}}{||\mathbf{r} - \mathbf{r}_O||^3}\right] \tag{6.4}$$

with $\mathbf{n} = (\mathbf{r} - \mathbf{r}_O)/||\mathbf{r} - \mathbf{r}_O||$. The distribution described by this expression is commonly referred to as the field of a dipole, see Fig. 6.2. For the special case of a sphere with homogeneous magnetization $\mathbf{M} = \frac{1}{2}[\mathbf{r} \times \mathbf{J}(\mathbf{r})]$, (6.4) represents the exact solution for all **r**.

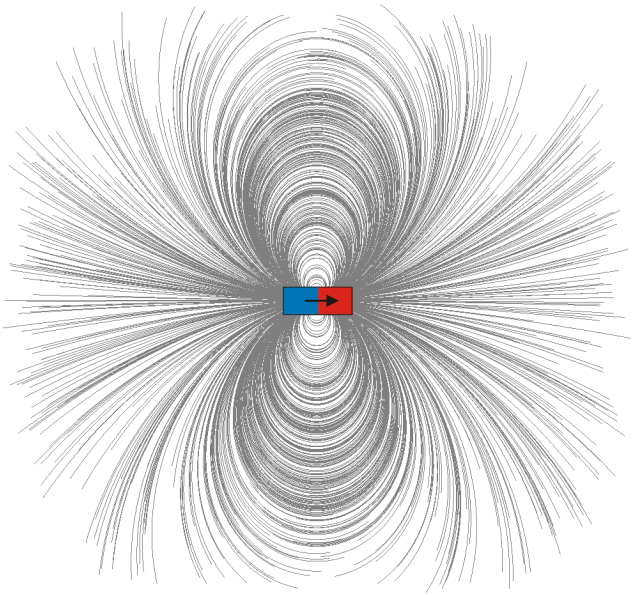


Figure 6.2: Field of a dipole cell with magnetization as indicated (*black arrow*).

Let us now consider an arbitrary magnetized object with volume V and homogeneous magnetization **M**. In order to model this object as an aggregate of magnetic dipoles, we subdivide its volume into N cells of equal size such that each of them carries a magnetic moment $\mathbf{m}_i = \mathbf{M}V/N$. Sampling an object with dipole cells is a straightforward process since it is sufficient to ensure that the cells are distributed evenly and that the sum of their moments corresponds to the

total magnetic moment $\mathbf{M}V$. In particular, a volume mesh is not required such that objects with complex geometries pose no problems. With this decomposition established, the total magnetic induction of the object can now be expressed as a linear superposition of the individual dipole fields,

$$\mathbf{B}(\mathbf{r}) = \frac{\mu_0}{4\pi} \sum_{i=1}^{N} \left[\frac{3\mathbf{n}_i(\mathbf{n}_i \cdot \mathbf{m}_i) - \mathbf{m}_i}{||\mathbf{r} - \mathbf{r}_i||^3} \right] \quad , \tag{6.5}$$

where $\mathbf{r}_i$ denote the positions of the cells and $\mathbf{n}_i = (\mathbf{r} - \mathbf{r}_i)/||\mathbf{r} - \mathbf{r}_i||$. If we assume the magnetization to be given, the exact solution for the total magnetic induction is recovered in the continuous limit $N \to \infty$. For finite N, Eq. (6.5) describes an approximation of the exact magnetic induction, whose accuracy is effectively controllable by the number of cells. Eq. (6.5) is only valid for distances larger than the spatial extent of the cells, which is V/N. However, with growing number of cells, the range of validity is increased to smaller and smaller distances to the object.

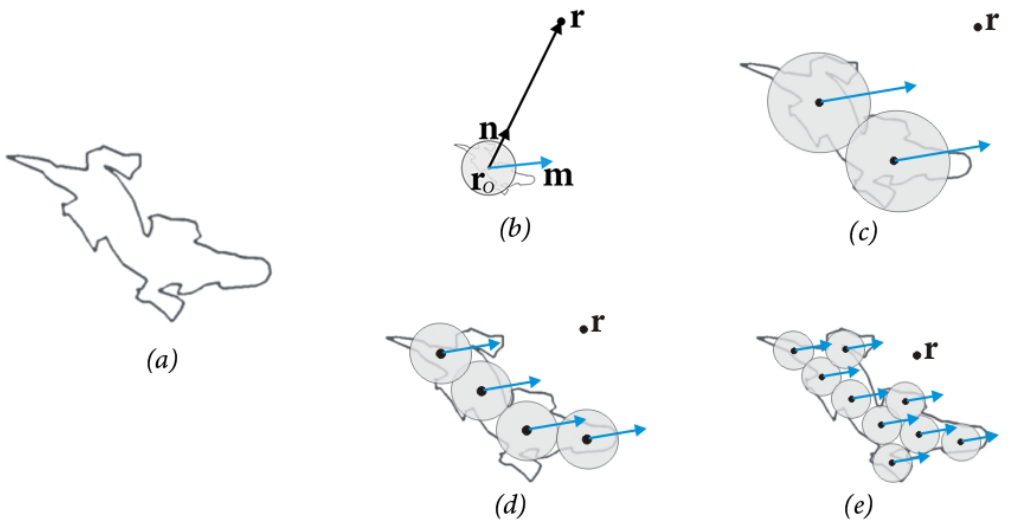


Figure 6.3: A dragon model (*a*) is sampled with dipole cells. For large distances $d = ||\mathbf{r} - \mathbf{r}_o||$, the field resembles a dipole field such that a single cell is sufficient (*b*). For smaller distances (*c*-*e*) the number of cells has to be increased in order to maintain the same level of accuracy.

The convergence of the total magnetic induction can be demonstrated on a simple numerical example. Fig. 6.4 shows the distribution of magnetic induction for a cuboidal test object with homogeneous magnetization using an increasing number of dipole cells. It can be seen that the distribution converges rapidly and that changes between the cases of 10×5 and 20×10 cells are already small. For more than 40×20 cells, only minimal differences in the immediate vicinity of the object can be observed. For distances in the range of the object's spatial extent, a few cells are already sufficient to obtain very good accuracy.

A question that arises in the context of this dipole approach is whether the case of contact is covered as well. If Eq. (6.4) is considered for a constant magnetic

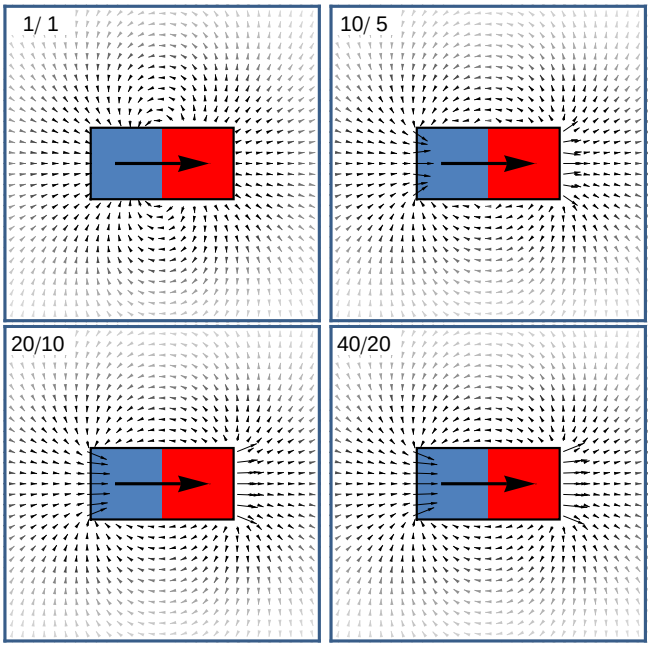


Figure 6.4: Magnetic induction $\mathbf{B}(\mathbf{r})$ for an object with homogeneous magnetization. The number of cells in x/y direction used for the calculation is indicated in the upper left corner of each figure. The shade of the vectors is proportional to their magnitude $||\mathbf{B}(\mathbf{r})||$.

moment, $\mathbf{B}(\mathbf{r})$ tends to infinity as the distance $d = ||\mathbf{r} - \mathbf{r}_O||$ goes to zero. Regardless of whether this expression is in good correspondence with the physical world, infinite field values are certainly not ideal for computer implementations. Theoretically, this problem can be avoided by appropriately increasing the cell density as a function of d. In particular, if the cell volume is made proportional to $1/d^3$, the magnetic induction $\mathbf{B}(\mathbf{r})$ remains finite as $d \to 0$. However, another problem arises since the number of cells now tends to infinity as contact is approached. We solve this problem by defining a maximum cell density and ensuring that two dipole cells cannot approach closer than a corresponding minimal cell diameter (see Sec. 6.3.2). This can also be interpreted as using a non-magnetic safety layer around magnetized objects, whose thickness is controlled by the maximum cell density. This view corresponds quite well with the plastic coated magnetic toys used in our examples (see Fig. 6.1).

6.2.2 Forces and Torques

If a current distribution $\mathbf{J}$ with volume V is exposed to an external magnetic field $\mathbf{B}$, it experiences forces and torques according to Ampère's law,

$$\mathbf{F} = \int_V \mathbf{J}(\mathbf{x}) \times \mathbf{B}(\mathbf{x}) \, dx \,, \tag{6.6}$$

$$\mathbf{T} = \int_V \mathbf{x} \times [\mathbf{J}(\mathbf{x}) \times \mathbf{B}(\mathbf{x})] \, dx \,. \tag{6.7}$$

The leading terms of these expressions can be obtained by a Taylor series expansion of the magnetic induction $\mathbf{B}$ around the center of the current distribution. The lowest-order nonvanishing terms in the expansion of the force and torque are

$$\mathbf{F} = \nabla(\mathbf{m} \cdot \mathbf{B}) \tag{6.8}$$

$$\mathbf{T} = \mathbf{m} \times \mathbf{B} \, . \tag{6.9}$$

From these expressions, we can determine the forces and torques acting between a pair of dipole cells. To this end, consider two dipole cells at positions $\mathbf{r}_l$ and $\mathbf{r}_k$ with magnetic moments $\mathbf{m}_l$ and $\mathbf{m}_k$, respectively. Inserting Eq. (6.4) into Eqs. (6.8) and (6.9) and letting $\mathbf{n}_{lk} = (\mathbf{r}_k - \mathbf{r}_l)/||\mathbf{r}_k - \mathbf{r}_l||$, we obtain explicit expressions for the force

$$\begin{aligned} \mathbf{F}_{kl} = \frac{\mu_0}{4\pi} \frac{1}{||\mathbf{r}_k - \mathbf{r}_l||^4} \Big[& -15\mathbf{n}_{lk}\big((\mathbf{m}_k \cdot \mathbf{n}_{lk})(\mathbf{m}_l \cdot \mathbf{n}_{lk})\big) \\ & + 3\mathbf{n}_{lk}(\mathbf{m}_k \cdot \mathbf{m}_l) + 3\big(\mathbf{m}_k(\mathbf{m}_l \cdot \mathbf{n}_{lk}) + \mathbf{m}_l(\mathbf{m}_k \cdot \mathbf{n}_{lk})\big)\Big] \end{aligned} \tag{6.10}$$

and the torque

$$\mathbf{T}_{kl} = \frac{\mu_0}{4\pi} \Big[\frac{3(\mathbf{m}_k \times \mathbf{n}_{lk})(\mathbf{m}_l \cdot \mathbf{n}_{lk}) - \mathbf{m}_k \times \mathbf{m}_l}{||\mathbf{r}_k - \mathbf{r}_l||^3} \Big] \tag{6.11}$$

acting on $\mathbf{m}_k$ due to the field of $\mathbf{m}_l$. Note that with increasing distance, force decreases more rapidly than torque, by a factor of $1/||\mathbf{r}_k - \mathbf{r}_i||$. For large distances, the interaction is therefore dominated by torque.

Instead of considering only the field of a single cell, we can also use Eq. (6.5) for the field of a magnetized object with N cells to obtain

$$\mathbf{F}_k = \sum_{i=1}^{N} \mathbf{F}_{kl} \, , \qquad \mathbf{T}_k = \sum_{i=1}^{N} \mathbf{T}_{kl} \, . \tag{6.12}$$

In order to compute the total force $\mathbf{F}$ and torque $\mathbf{T}$ acting on an object composed of M dipole moments, (6.10) and (6.11) have to be evaluated for all cells ($k = 1 \ldots M$) and summed up appropriately. For the total force, the positions of the cells $\mathbf{r}_k$ relative to the center of mass $\mathbf{R}$ of the object have to be respected in the summation. Letting $\mathbf{T}_m$ denote the net mechanic torque caused by the magnetic dipole forces $\mathbf{F}_k$, we have

$$\mathbf{F} = \sum_{k=1}^{M} \frac{\mathbf{F}_k \cdot (\mathbf{r}_k - \mathbf{R})}{||(\mathbf{r}_k - \mathbf{R})||} \cdot \frac{(\mathbf{r}_k - \mathbf{R})}{||(\mathbf{r}_k - \mathbf{R})||} \, , \qquad \mathbf{T}_m = \sum_{k=1}^{M} (\mathbf{r}_k - \mathbf{R}) \times \mathbf{F}_k \, , \tag{6.13}$$

and the total torque follows as

$$\mathbf{T} = \sum_{k=1}^{M} \mathbf{T}_k + \mathbf{T}_m \, . \tag{6.14}$$

The next section gives a geometric interpretation of all terms involved in these expressions and also sketches how to prove momentum conservation. But before we proceed to this subject, a brief comparison to the equivalent magnetic dipole

method as described in [DMMR01, Del01] will be made. The latter defines the total force $\mathbf{F}_K$[2] on a magnetized object with domain V as

$$\mathbf{F}_K = \int_V \mathbf{M} \cdot \nabla \mathbf{B}_{ext} \, dV \, , \tag{6.15}$$

where $\mathbf{M}$ describes the continuous distribution of dipole moments and $\mathbf{B}_{ext}$ denotes the magnetic induction due to an external source. Considering this external source as an aggregate of dipoles, it is obvious that the expression described by Eq. (6.10) converges towards that of Eq. (6.15) as the number of cells, N and M, tends to infinity. Analogous observations can be made for the torque. Hence, Eqs. (6.10) and (6.11) are consistent and exact in the continuous limit, but accurate and reliable results are already obtained for a small number of cells per object. This robustness with respect to mesh refinement has also been observed by Delfino et al. [DMMR01], who investigated the accuracy of force computations with the equivalent magnetic dipole method in combination with finite element solutions for the external field.

6.2.3 Interpretation and Momentum Conservation

Interpretation It is insightful to give geometric interpretation to the different terms appearing in the expressions for forces (6.10) and torques (6.11) acting between pairs of dipole cells. To this end, we investigate six exemplary arrangements of two bar magnets (Fig. 6.5), which are modeled as simple magnetic dipoles with a single cell. Only the force and torque terms acting on the second bar magnet are discussed in the following. Analogous observations for the first magnet follow by interchanging $\mathbf{m}_1$ and $\mathbf{m}_2$ and setting $\mathbf{n}_{12} = -\mathbf{n}_{21}$.

In the first arrangement A1, the two bar magnets are aligned in parallel and positioned side by side. The first term of the dipole force (6.10) vanishes since both $(\mathbf{m}_2 \cdot \mathbf{n}_{12})$ and $(\mathbf{m}_1 \cdot \mathbf{n}_{12})$ evaluate to zero, but the second term contributes proportionally to $3\mathbf{n}_{12}(\mathbf{m}_2 \cdot \mathbf{m}_1)$ (yellow arrow in the sketch, labeled 2). The third term of (6.10) vanishes for the same reason as the first one. The first term of the dipole torque (6.11) does not contribute since $(\mathbf{m}_1 \cdot \mathbf{n}_{12})$ is zero, and the second term of (6.11) vanishes since $\mathbf{m}_2 \times \mathbf{m}_1 = 0$. Accordingly, the only relevant contribution is given by the second term of Eq. (6.10) and the total force is repulsive.

The second arrangement A2 is analogous to A1, but this time with an antiparallel alignment. Consequently, the total force is attractive.

This is also the case in the third arrangement B1, in which all three terms (labeled 1 to 3) of Eq. (6.10) contribute. The first term has the largest prefactor and dominates the total force. The situation in the fourth arrangement B2 is the same as in B1, but with $\mathbf{m}_2$ reversed. Accordingly, all contributions change sign and the total force is repulsive. In both B1 and B2, the torque vanishes since $(\mathbf{m}_2 \times \mathbf{n}_{12}) = 0$ and $(\mathbf{m}_2 \times \mathbf{m}_1) = 0$.

The torque comes into play in the last two arrangements C1 and C2. Here, only the third term of the force (6.10) contributes. The first term of the torque (6.11), which is proportional to $3(\mathbf{m}_2 \times \mathbf{n}_{12})(\mathbf{m}_1 \cdot \mathbf{n}_{12})$, acts only on $\mathbf{m}_1$ (labeled 4). The second term of (6.11) contributes for both $\mathbf{m}_1$ and $\mathbf{m}_2$ and is proportional

[2] this is also known as Kelvin's formula

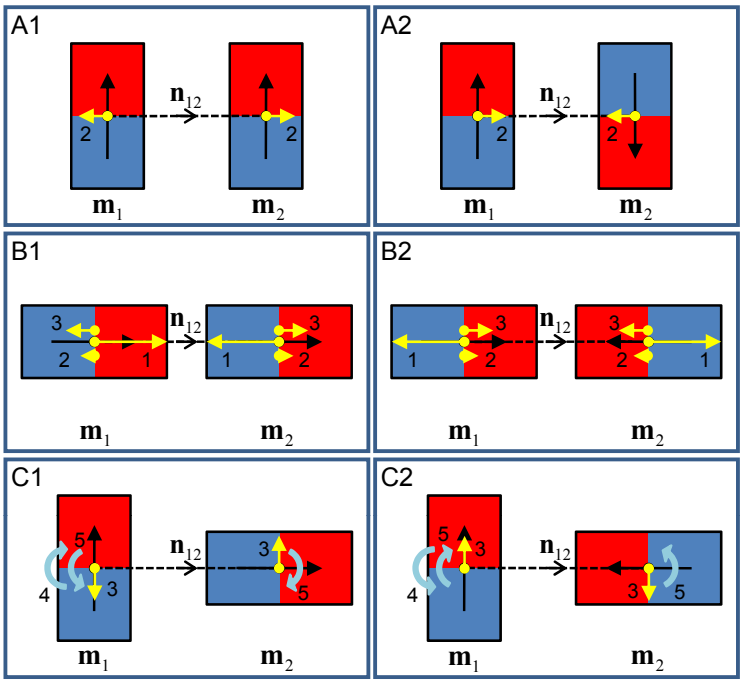


Figure 6.5: Six exemplary arrangements of two bar magnets with forces and torques depicted by yellow and blue arrows. Black arrows indicate the magnetic dipole moments $\mathbf{m}_1$ and $\mathbf{m}_2$, "north" and "south" poles of the magnets are shown in red and blue.

to $-(\mathbf{m}_2 \times \mathbf{m}_1)$ (labeled 5). The reversion of $\mathbf{m}_2$ from C1 to C2 leads to a change in sign for all contributions. In C1, the total torque tends to align the two bar magnets in a way similar to the configuration of B1. In C2, the total torque also tends to align the two bar magnets, but in the opposite direction.

Momentum Conservation Conservation of momentum is an inherent property of conservative physical systems, but it is not obvious that an approximate description exhibits the same behavior. In the present case, however, the approximations for the magnetic induction, forces and torques are formulated in such a way that both linear and angular momentum are conserved. In order to prove conservation, it has to be shown that both the total force and the total torque (summed over all cells of all objects) vanish. For the total force, this can easily be seen from (6.10) by interchanging $\mathbf{m}_k$ and $\mathbf{m}_i$ as well as their positions. Then, the force acting on $\mathbf{m}_k$ is exactly the negative of the force acting on $\mathbf{m}_i$ and the sum of the two vanishes. For the total torque, the derivation involves both forces and torques since the third term of Eq. (6.10) as well as Eq. (6.11) contribute. In analogy to Eq. (6.13), a correct combination of these contributions via $\mathbf{R} \times \mathbf{F} + \mathbf{T}$ leads to a vanishing total torque.

6.2.4 Magnetic Materials

We will now turn to the physical description of magnetic materials, i.e., the constitutive relations linking the magnetic induction $\mathbf{B}$ to the magnetic field $\mathbf{H}$. The exposition is structured according to the distinction between permanently magnetized objects and those with induced magnetization.

Permanent Magnetization Permanent magnets are typically ferromagnets with a strong remanence or residual magnetism. More precisely, their constitutive relation is given by a hysteresis curve $\mathbf{B}[\mathbf{H}]$ which exhibits a large remanent value of $\mathbf{B}$ for vanishing $\mathbf{H}$. In other words, a strong permanent magnetization is observed even in the absence of magnetizing fields. The field used for exciting the remanent magnetization is typically much stronger than the fields produced by permanent magnets. Consequently, the magnetization of a permanent magnet remains largely unaffected by external fields due to other permanent magnets. Accordingly, we assume the magnetization of permanent magnets to be constant (*hard ferromagnets*).

Induced Magnetization The second group of materials exhibits a magnetization only in the presence of an external magnetic field. This group includes diamagnets, paramagnets as well as ferromagnets with a weak remanence. Superconductors will also be treated within the scope of this group.

For isotropic[3] diamagnets and paramagnets, the constitutive relation can be written in a linear form as

$$\mathbf{B} = \mu_0(\mathbf{H} + \mathbf{M}) = \mu_0(1 + \chi)\mathbf{H}\,, \tag{6.16}$$

with χ denoting the magnetic susceptibility of the material. In the case of a diamagnet, we have $\chi < 0$, whereas for paramagnets $\chi > 0$. For the most common diamagnetic and paramagnetic materials, however, the absolute value of the susceptibility is of the order of 10^{-6} and the resulting magnetization as well as the forces and torques are very weak.

As mentioned above, ferromagnets have to be described by a hysteresis curve that puts $\mathbf{B}$ and $\mathbf{H}$ into relation. For the special case of ferromagnets with a weak remanence (*soft ferromagnets*) subject to weak external magnetic fields, a linear approximation in the form of (6.16) with $\chi > 0$ is, however, adequate. The magnetic susceptibility of soft ferromagnets varies by several orders of magnitude for different materials. Steel, for example, exhibits a value of about $\chi \simeq 700$ whereas $\chi \simeq 20,000$ for mu-metal, which is a nickel-iron alloy.

In general, the magnetostatic Maxwell equations (6.1) and (6.2) have to be solved together with the constitutive relation, i.e., (6.16) in the present case. For arbitrary geometries, this is a very complicated task because of the boundary conditions that have to be respected, e.g., at the interface of two media [Jac99]. Computationally most efficient are explicit analytical solutions, but they can only be obtained for highly symmetric situations. Fortunately, modeling magnetized objects as aggregates of dipole cells and approximately assuming the cells to be

[3] Magnetic materials are said to be *isotropic* if the resulting magnetic induction $\mathbf{B}$ is parallel to the magnetizing field $\mathbf{H}$.

spherical, an explicit solution can be derived. If we additionally neglect the coupling between the cells[4], we obtain an expression for the induced magnetization $\mathbf{M}_i$ of a cell located at position $\mathbf{r}_i$ as

$$\mathbf{M}_i = \frac{3}{\mu_0}\frac{\chi}{1+\chi}\mathbf{B}(\mathbf{r}_i) \ . \tag{6.17}$$

This model is only approximate and certainly less accurate than solving Maxwell's equations together with the constitutive relations. However, it is sufficient to reproduce the salient macroscopic properties and, as an important advantage, allows for a very efficient implementation of magnetic interaction: we first use (6.5) to compute the magnetic induction due to permanently magnetized objects and then determine the induced magnetization of diamagnets, paramagnets and soft ferromagnets using (6.17). The resulting final magnetic induction is used to compute forces and torques according to Eqs. (6.10) and (6.11).

It can be seen from Eq. (6.17) that the sign of the magnetic susceptibility χ determines the general behavior of the material. In the case of a positive (negative) value, the induced magnetization is oriented parallel (anti-parallel) to the magnetic induction. Accordingly, a paramagnet is attracted by a permanent magnet, whereas a diamagnet is repelled. Soft ferromagnets with $\chi > 0$ behave like paramagnets, but the resulting force and torque are much stronger due to the higher value of χ.

Superconductors expel any external magnetic field from their interior[5], which is achieved by screening currents flowing on the surface of the superconducting material. In the interior of the superconductor, these screening currents generate a magnetic field which exactly balances the external one. In a simplified picture, the magnetic induction thus vanishes completely inside the superconductor and upon inserting $\mathbf{B} = 0$ into Eq. (6.2), we find $\mathbf{M} = -\mathbf{H}$ (*perfect diamagnetism*). If we additionally approximate the magnetic field $\mathbf{H}$ by the vacuum field, we have $\mathbf{H} = \mathbf{B}/\mu_0$ and thus

$$\mathbf{M}_i = -\frac{1}{\mu_0}\mathbf{B}(\mathbf{r}_i) \tag{6.18}$$

for every cell of the superconductor. Because of the strong induced magnetization which is oriented anti-parallel to the magnetic induction, superconductors are strongly repelled from permanent magnets.

6.3 Implementation

This section describes implementational aspects of our method including the computation of magnetic fields, forces and torques as well as the adaptive cell hierarchy.

[4] i.e, we assume that the field due to a cell with induced magnetization has no influence on the magnetization of other cells

[5] a phenomenon called Meißner-Ochsenfeld effect

6.3.1 Algorithm

Algorithm 6.1 provides an overview of the steps involved in the computation of magnetic forces and torques. The magnetization of induced magnets changes according to the magnetic field in their surroundings and has to be recomputed in every time step. This amounts to evaluating the magnetic field of all permanent magnets in the scene at the induced magnets' cell positions (l.3). It should be stressed that the field is only evaluated where it is required for computations. In particular, we do not employ a spatial grid for this purpose[6]. The following code (ll.6-16) computes the magnetic interaction between all magnets in the scene. Note that the bound for the second loop (l.7) is different due to the symmetry of magnetic interaction. For every pair of magnets, we compute force and torque exerted by all cells of one object onto all cells of the other object and vice versa according to Eqs. (6.10) and (6.11).

Algorithm 6.1 Magnetic force and torque computation

```
 1: //Compute magnetization for induced magnets
 2: for all induced magnets do
 3:   computeInducedMagnetization();
 4: end for
 5: //Compute magnetic force and torque
 6: for i = 1 to n_mag do
 7:   for j = i to n_mag do
 8:     for k = 1 to n_cells,i do
 9:       for l = 1 to n_cells,j do
10:         c_k=cell(k), c_l = cell(l);
11:         applyForce(i,c_k,c_l), applyTorque(i,c_k, c_l);
12:         applyForce(j,c_l,c_k), applyTorque(j,c_l, c_k);
13:       end for
14:     end for
15:   end for
16: end for
```

Once forces and torques between a pair of dipoles are computed, they can either be fed directly into the rigid body dynamics solver (ll.11-12) or be accumulated first. The latter is typically more efficient, especially in a parallel implementation where thread-safety of the rigid body dynamics code might be an issue. The actual parallelization is simple and efficient since all operations for one iteration are independent of other iterations.

For practical reasons, the decomposition of objects into dipole cells is constructed in such a way that the distance between two cells can never fall below the cell diameter, even in the case of contact. This is not a fundamental restriction since in the limit of an infinite number of cells the model is still accurate (see Sec. 6.2.1). For reasonable magnetization densities, the magnetic forces are thus safely bounded such that explicit time stepping provides sufficient stability. However, we cannot rely on the collision code to always guarantee an intersection-free state and cells may in fact become arbitrarily close. This problem can be dealt with

[6] except for visualizing magnetic field lines

by limiting force and torque to the maximum values corresponding to the closest possible distance between two cells without intersections.

The run time of algorithm 6.1 scales quadratically with respect to the total number of cells in the scene. An adaptive approach is therefore mandatory in order to decrease the average computational costs and still guarantee sufficient resolution where necessary. As explained subsequently, we can leverage the controllable accuracy of magnetic induction (6.5), force (6.10) and torque (6.11) for constructing an adaptive multi-resolution cell hierarchy.

6.3.2 Adaptive Refinement

As indicated in Sec. 6.2, a higher cell density increases the approximation quality of magnetic fields, forces and torques. For complex scenes, however, using a uniformly high cell resolution is prohibitively expensive and we therefore resort to adaptivity. The central observation for our adaptive strategy is the fact that the magnetic induction decays very rapidly away from its source. More precisely, the magnetic induction $\mathbf{B}(r)$ of a dipole is proportional to $1/r^3$, where r denotes the distance to the dipole. If we prescribe a sufficient ratio between the magnitude $||\mathbf{B}||$ and the spatial cell density ρ as $||\mathbf{B}||/\rho = c_0 = const.$ we obtain a means of adapting the number of cells in a region of space to the local magnetic field. The remainder of this section describes the adaptive scheme that builds upon this measure.

Cell Hierarchy The basis for the adaptive refinement scheme is formed by a multi-resolution hierarchy of cells, which bears some similarity to the octree structure used by Barnes and Hut [BH86]. Related is also the work of Klein and Ertl [KE04], who use a spatial subdivision scheme to accelerate the computation of potential energies for distant particle pairs. However, instead of partitioning space we construct hierarchies on a per-object basis. For a given object, the construction of the hierarchy starts with a single cell at level zero, which corresponds to its bounding box. Although a cell itself has only volume and no particular shape, we associate with it the bounding box of the geometry that it represents and define its center as the position of the cell. The cells on level $i+1$ are obtained through bisection of the cells on level i along their longest axis. We prefer bisection over octasection as it leads to slower growth in the number of cells per level. If we find that a newly created cell lies completely outside the object, it is immediately discarded. The process stops when the cell density of the finest level reaches a prescribed value ρ_0. An indication of an appropriate value for ρ_0 can be obtained from Fig. 6.7 (e.g., eight cells per cubic centimeter), but in general, this value has to be set in relation to the dimensions of the objects. Subsequently, we eliminate all those leaf cells whose positions lie outside the object or which are closer to the object boundary than half their diameter. This process is applied to each level in order to remove all empty cells without children. Finally, we rescale the magnetizations of all child cells evenly such that their sum matches the total magnetization of the corresponding parent cells. This step is necessary to guarantee that force and torque computations are consistent among the levels: for sufficiently large distances, a dipole will thus experience the same force and torque from any level

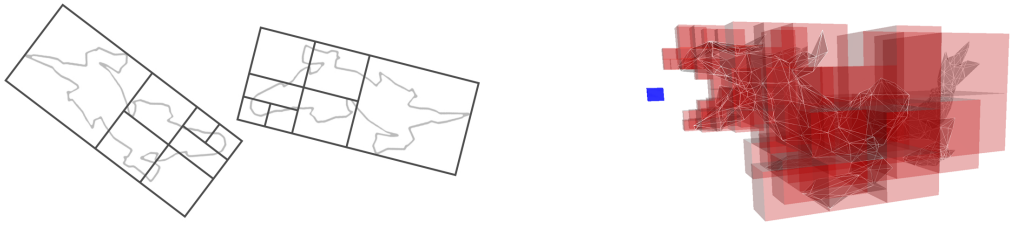


Figure 6.6: *Left*: schematic view of proximity-based adaptive refinement in two dimensions. *Right*: cell front for a dragon model with respect to a given point in space (*blue box*).

of a magnetized object.

Cell Front As mentioned in Sec. 6.3.1, magnetic forces, torques and fields are always computed for pairs of magnets. Given such a pair, we have to select a resolution level based on the geometric distance between the two objects. Using a single hierarchy level would mean fixing the cell density throughout the entire objects, which is inefficient if only small regions are close, but large parts are far away from each other (see Fig. 6.6, left). Instead, we compute an adapted cell sampling or *cell front* for each of the two objects as follows: starting with the top level cell of one object, we first determine the distance r_c that would be sufficiently large for the current cell density $\rho_c = 1/v_c$, with v_c being the object's volume. By construction we have

$$\frac{\rho_0}{\rho_c} = \frac{r_c^3}{r_0^3} \qquad \text{such that} \qquad r_c = \sqrt[3]{\frac{\rho_0}{\rho_c}} r_0 \; . \tag{6.19}$$

We then determine whether the geometric distance d of the cell to the other object is smaller than r_c. To accelerate this process, we first check the corresponding bounding spheres for overlap and, if they intersect, compute d using a GJK-test [GJK88] on the oriented bounding boxes. The current cell density is sufficient if $d >= r_c$. Otherwise, we apply the same process recursively to all children of the current cell until a sufficient density is reached. This process is applied to both objects, yielding cell fronts which reflect the continuous geometric distance between the objects (see Fig. 6.6). Note that the hierarchy needs only be constructed once at the beginning of the simulation. Additionally, Algorithm 6.1 requires only minor changes to ll.8-9 in order to support adaptive refinement.

6.3.3 Numerical Validation

The method presented in this chapter is intended for physically-based simulations and we provide several examples in Sec. 6.4. However, we also explicitly verified its validity and accuracy with respect to force computations and momentum conservation.

Analytical Comparison In order to assess the accuracy of the magnetic forces, we applied our method to a test configuration consisting of two cube-shaped fer-

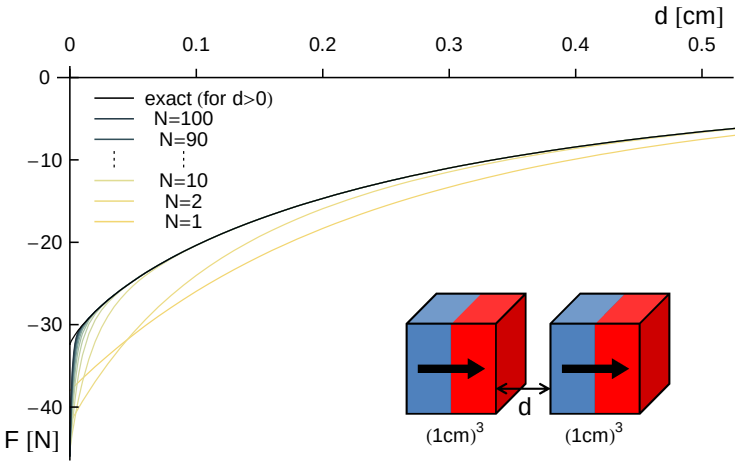


Figure 6.7: Comparison of the total force calculated using our method with exact results for a test configuration consisting of two ferromagnetic cubes with homogeneous magnetization. With increasing number of cells per object, (given by N^3 with N as indicated), the results for the force converge to the exact solution.

romagnets with volume $V = (1\text{cm})^3$ and a homogeneous remanent magnetic induction $\mathbf{B}_r$ of 1 Tesla (see Fig. 6.7). Using Eq. (6.2), the corresponding magnetization is found to be $\mathbf{M} = \mathbf{B}_r/\mu_0$. This test case has been used by de Medeiros et al. [dMRM98] for comparing several alternative force calculation methods and an analytical solution has been derived by Akoun and Yonnet [AY84].

The results shown in Fig. 6.7 were obtained using N^3 uniformly distributed cells for each cube. For the fixed distance of $d = 0.5\,\text{cm}$ used in [dMRM98], our method reproduces the correct result for the attractive force already with a very low number of cells. For lower distances, the results of our method converge to the exact solution as the number of cells is increased.

The results for the total force as given in Fig. 6.7 further emphasize the usefulness of adaptivity: for close proximity a high cell resolution is mandatory in order to obtain good accuracy. However, the number of cells necessary for maintaining good approximation quality rapidly decreases with growing inter-object distance.

Momentum Conservation The conservation of linear and angular momentum is an important property of the approach presented in this chapter. Additionally, it also provides a simple way to check the implementation. For testing purposes, linear and angular momentum are readily computed during the simulation. If the implementation is correct, the sum of both quantities over all objects has to be time-invariant. We verified this for our code on a number of simulations and obtained very good conservation behavior for the frictionless and non-contact case. During and after contact situations, the conservation deteriorates even for completely elastic collisions. However, this behavior is fully attributable to the rigid body dynamics solver and not a defect of our method.

6.4 Results

This section presents examples of magnetic interactions obtained with the method presented in this chapter. As the basis of our implementation, we used the freely available rigid body dynamics library ODE [Smi06]. Many of our examples use *toy magnets*, which consist of a plastic-coated cylinder with two small permanent magnets at the ends (see Figs. 6.1, 6.8 and 6.9). Simulated toy magnets can easily be compared to their real-world counterparts using simple experiments and offer a great potential for discovering a variety of magnetic interactions. The magnetic field of a toy magnet is visualized in Fig. 6.1.

6.4.1 Examples

The first test case (Example 6.1) demonstrates the effect of inhomogeneous induced magnetization. As shown in Fig. 6.8, a magnetized sphere is held in contact with a toy magnet, while a second toy magnet is placed at some distance.

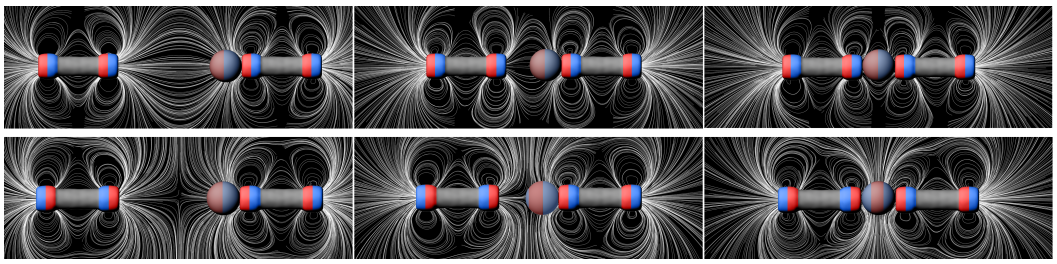


Figure 6.8: Induced inhomogeneous magnetization illustrated on a soft-ferromagnetic sphere and two toy magnets with parallel (*top row*) and anti-parallel (*bottom row*) magnetizations.

In the first case, the toy magnets have the same magnetization and are therefore attracted uniformly to each other (Fig. 6.8, top). Consequently, the sphere is magnetized homogeneously (identical direction) according to the external field and attracts the left toy magnet as well. In the second and more interesting case, the toy magnets have opposite magnetizations leading to a point of zero resulting magnetic field, clearly distinguishable in the left-most figure of the bottom row. Initially, the sphere assumes the magnetization of the right toy magnet, to which it is attached. As the left toy magnet is gradually moved to the right, its magnetic field starts to induce a corresponding magnetization in the sphere, resulting in a directionally inhomogeneous magnetization. However, the left toy magnet is still repelled from the right one. Forcing it to move further rightwards, the repelling forces increase but, at the same time, the attracting field induced in the sphere grows up to the point, where the resulting attraction forces exceed the repelling forces and the toy magnet snaps to the sphere. This fascinating effect, which can be described as passing a potential barrier, is clearly perceptible in reality and faithfully reproduced by our method. Note that although the final geometries of the top and bottom sequences are identical, the resulting fields exhibit subtle local differences.

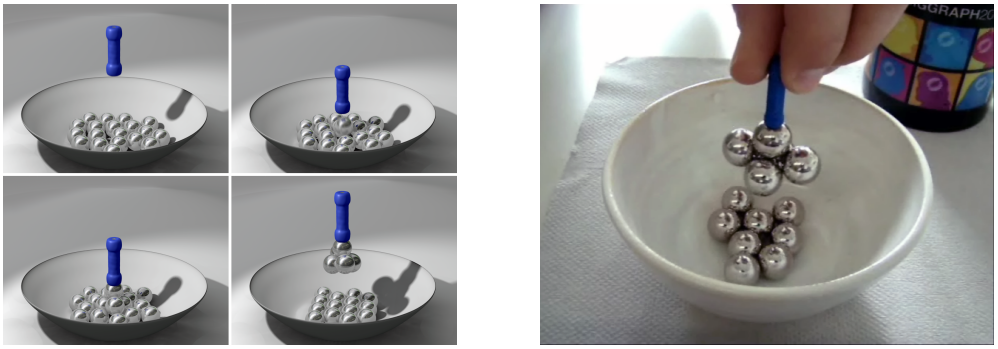


Figure 6.9: *Left*: snapshots of an interactive simulation in which soft-ferromagnetic spheres are lifted by a permanent magnet. *Right*: corresponding real-world experiment.

Example 6.2 is another illustration of the complex behavior obtained with multiple induced magnets. The left part of Fig. 6.9 shows a sequence of four images taken from a simulation of a permanent magnet lifting several soft-ferromagnetic spheres out of a bowl. As the permanent magnet is moved downwards it starts to induce a magnetization in the spheres. At a certain distance, one of the spheres is picked up by the permanent magnet (see upper two images of Fig. 6.9). Subsequently, the other spheres are further drawn towards the magnet and are lifted as well. Finally, five spheres are attached to the permanent magnet and lifted above the bowl (lower two figures). In the final position, only one of the spheres is directly attached to the permanent magnet whereas the other four spheres are hanging below the first one, forming a symmetric configuration. As can be seen in the right part of Fig. 6.9, this example is in good correspondence with the behavior observed in a real-world experiment.

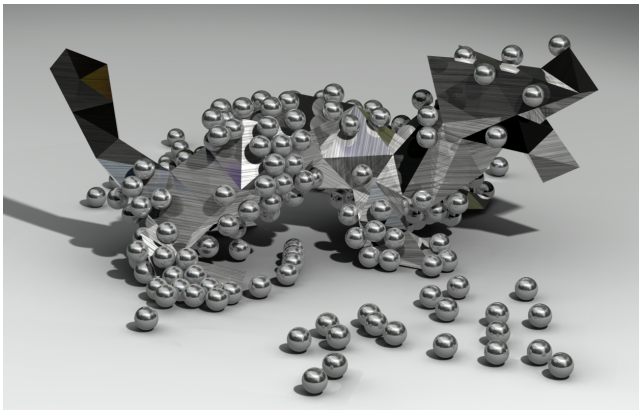


Figure 6.10: A permanently magnetized simple dragon model is exposed to a downpour of soft-ferromagnetic spheres. Some of the spheres are held on the surface of the dragon due to magnetic attraction.

Two further examples are intended to demonstrate that our method is capable of handling a large number of objects as well as magnets with arbitrary non-convex geometry. Fig. 6.10 shows an image from Example 6.3 in which 250

soft-ferromagnetic spheres are dropped onto a permanently magnetized dragon model. As soon as they approach the dragon, a magnetization is induced in the spheres and they start to interact magnetically with each other and the dragon.

Fig. 6.11 shows a sequence of images from Example 6.4 in which a rectangular permanent magnet lifts four soft-ferromagnetic characters out of a pool of non-magnetic spheres.

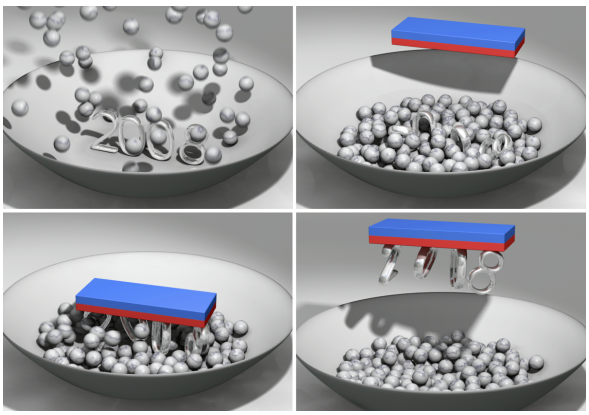


Figure 6.11: A strong permanent magnet lifts four soft-ferromagnetic characters.

Our method also offers the possibility to experiment with effects of magnetism that are difficult to produce or access in reality such as superconductivity. If a superconductor is placed in the vicinity of a hard ferromagnet, the induced magnetization within the superconductor is oriented anti-parallel to the field of the ferromagnet. According to Eq. (6.18), the induced magnetization exactly balances the external field and can thus be of considerable strength. With a sufficiently strong magnetization of the ferromagnet the repelling forces on the superconductor can be large enough to overcome gravity.

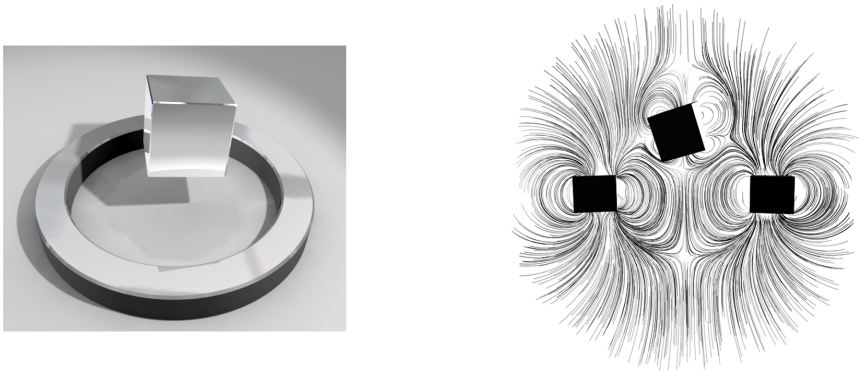


Figure 6.12: Rendered frame and field line visualization for a superconducting cube levitating above a ferromagnetic ring.

This effect is reproduced in the animation of Example 6.5, shown in Fig. 6.12, which consists of a superconducting cube levitating above a ferromagnetic ring. The toroidal shape of the ferromagnet leads to an equilibrium position above the

center of the ring for the superconducting cube. The corresponding field line plot shows that the ferromagnetic ring is magnetized in the vertical direction. In two points above and below the center of the ring, the magnetic field vanishes. Since the superconductor is repelled from the field, positions close to these two points are energetically favorable. Once put into the upper of the two, the superconducting cube remains in this stable position. It can also be observed that the field lines are repelled from the interior of the superconductor and literally flow around it[7].

Apart from dynamic simulation, our method can also be used as a tool for computing magnetic fields for arbitrary objects (see Fig. 6.13). This is especially interesting for exploring the relation between field structure and geometric shape for objects which are difficult to manufacture.

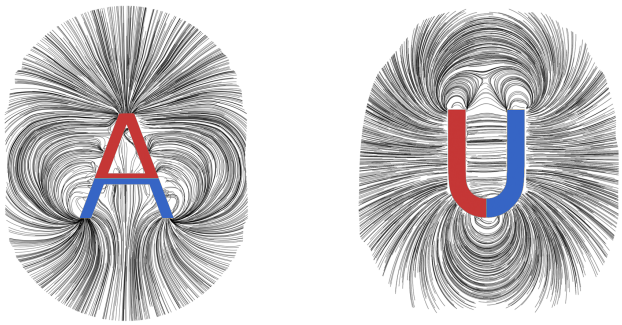


Figure 6.13: Detailed visualizations of magnetic fields created with a standard streamline technique.

6.4.2 Performance

The computational impact of our method is determined by the number of cells used to compute magnetic forces and torques. Compared to the worst case performance, which scales quadratically with the number of cells, the average run time is greatly reduced using the adaptive sampling described in section 6.3.2. The benefit of adaptivity depends, however, on the scene under consideration (see Table 6.1) and is generally not constant over time. For the examples described above, the average reduction in computation time due to adaptivity ranges from 1.3 for Example 6.1 to 11.1 for Example 6.3.

[7] this is the Meißner-Ochsenfeld effect mentioned above

Example	Figure	#cells	T_{mag}	T_{rbd}	T_{tot}	Gain	fps
6.1	6.8	48	0.29	0.09	0.38	1.30	65
6.2	6.9	256	3.9	0.25	4.16	3.76	6
6.3 (*1*)	6.10	1824	53.1	2.76	55.85	6.72	0.45
6.3 (*2*)	6.10	3628	57.6	2.92	60.5	11.1	0.41
6.4	6.11	128	1.65	2.39	4.05	1.97	6.2
6.5	6.12	142	0.33	0.02	0.35	5.97	71.4

Table 6.1: Average computation times in milliseconds for a single time step of $0.001s$ on an Intel Core2Duo 2.4GHz CPU (using only one core). Computation times for magnetic forces (T_{mag}) and time spent in the RBD code (T_{rbd}) are listed separately. *#cells* refers to the total number of cells in the scene and *Gain* denotes the speedup for magnetic force computations due to adaptivity, which was used in all examples. The setup for Examples 6.3 (*1*) and (*2*) is identical except for the number of cells.

Table 6.1 summarizes the performance for the examples presented in this section, showing total computation times and the time spent on computing magnetic forces and torques. To facilitate interpretation, computation times are provided for the sequential version of Algorithm 6.1. However, many parts of the method lend themselves to easy parallelization strategies and a prototype implementation showed nearly optimal speedup on a workstation with four CPUs. As expected, computing the magnetic interaction takes up most of the simulation time. Depending on the nature of the individual scenes, however, rigid body dynamics and especially collision handling can also require a considerable part of the total computation time. Nevertheless, simulations with dozens of magnets and up to several hundreds of cells run at interactive rates.

6.5 Conclusion

Summary This chapter introduced a computational method for modeling magnetic interactions in rigid body simulations. Its central idea is to compute forces and torques between magnetized objects from pairwise interactions between magnetic dipole cells. Decomposing magnets into aggregates of dipole cells directly leads to a discrete approach such that expensive numerical discretization techniques are avoided. We have advocated a symmetric approach in which dipole approximations are used for magnetic fields as well as for the resulting forces and torques. This allowed us to derive closed-form expressions for forces and torques in such a way that linear and angular momentum are automatically conserved in the discrete setting. Furthermore, we proposed an adaptive hierarchical sampling scheme that exploits the rapid decay of the magnetic field in order to accelerate computations.

The qualitative performance of this method has been investigated on a number of practical animations, which also demonstrated its ability to reproduce a wide range of macroscopic magnetic effects. Finally, the quantitative performance was assessed in terms of computation times, which indicated good efficiency suitable for interactive applications.

Limitations and Future Directions Perhaps the most fundamental difference between the method presented in this chapter and approaches based on a numerical solution of Maxwell's equations is the fact that constitutive relations are not accurately taken into account. The simplifying assumptions on the material behavior (see Sec. 6.2.4) were made deliberately in order to increase computational efficiency. While these approximations are probably tolerable for hard ferromagnets, they are too inaccurate for soft ferromagnets in order to be useful in engineering applications. In our approach, induced magnets exert forces on each other but their magnetizations are only influenced by the fields of permanent magnets and not by the fields due to other induced magnets. A correct treatment would be to compute a steady-state solution by solving Maxwell's equations together with constitutive relations and appropriate boundary conditions at media interfaces. In the context of our approach, accuracy could be improved by reiterating the magnetic field computation, including induced fields, until a steady-state is reached. Our experiments in this direction indicated that additional effort is necessary to

ensure convergence, which seems very sensitive to the strength of the magnetic field as well as the magnetic susceptibility of the material.

The approach described in this chapter conserves linear and angular momenta, but this does not imply energy conservation. A theoretical analysis of the latter seems difficult because of the energy accumulated in the magnetic field. Since the rigid body solver used in our implementation does not conserve energy, we did not try to assess this behavior experimentally.

An interesting extension could also be to include electric currents as sources of magnetic fields and to take into account corresponding forces and torques on these currents. In this way, it would be possible to model a virtual electric motor and simulate the interplay of electric, mechanical and magnetic parts.

Appendix A

Derivation of Finite Element Forces

This Appendix provides a brief derivation of the nodal forces resulting from the finite element membrane model described in Chapter 2. For a more comprehensive derivation of the governing equations of nonlinear continuum mechanics and their finite element discretization, we refer to the textbooks by Bathe [Bat96] and Bonet and Wood [BW97]. We start with equilibrium considerations in the continuous setting that will give rise to the virtual work equation.

A.1 Virtual Work Equation

As in Sec. 2.1, we let $\mathcal{B}$ denote a deformable body that is subjected to surface tractions $\mathbf{s}$ per unit area and body forces $\mathbf{b}$ per unit volume. Assuming that $\mathcal{B}$ is in static equilibrium, we have

$$\int_v \mathbf{b}\, dv + \int_{\partial v} \mathbf{s}\, da = 0\,, \tag{A.1}$$

where v denotes the deformed volume of $\mathcal{B}$ and ∂v its boundary. According to Eq. (2.14), the traction forces cause stress on the boundary, $\mathbf{s} = \sigma\mathbf{n}$, where $\mathbf{n}$ is the outward normal on ∂v. Using this in (A.1) and transforming the surface integral according to the divergence theorem, we obtain

$$\int_v \mathbf{b}\, dv + \int_{\partial v} \sigma\mathbf{n}\, da = \int_v \operatorname{div}\, \sigma + \mathbf{b}\, dv = 0\,. \tag{A.2}$$

Since this statement must hold for any enclosed region inside the body $\mathcal{B}$, the point-wise equilibrium equations of continuum mechanics are recovered as

$$\operatorname{div}\, \sigma + \mathbf{b} = 0\,. \tag{A.3}$$

This *strong form* of equilibrium condition is not an adequate starting point for numerical treatment. We begin the transformation to a more accessible *weak form* by multiplying Eq. (A.2) by a *test function* $\delta\mathbf{u} \in \mathbb{R}^3$ to obtain

$$\int_v (\delta\mathbf{u}^t \operatorname{div}\, \sigma + \delta\mathbf{u}^t\mathbf{b})\, dv = 0\,. \tag{A.4}$$

Here, $\delta\mathbf{u}$ is arbitrary but has to be sufficiently smooth[1] and compatible with the boundary conditions. We first transform[2]

$$\delta\mathbf{u}^t \text{div}\, \sigma = \text{div}\, (\sigma\delta\mathbf{u}) - \sigma : \nabla\delta\mathbf{u} \tag{A.5}$$

and then apply the divergence theorem to the first term on the right hand side to obtain

$$\int_v \text{div}\, (\sigma\delta\mathbf{u})\, dv = \int_{\partial v} (\sigma\delta\mathbf{u}) \cdot \mathbf{n}\, dv = \int_{\partial v} \delta\mathbf{u}^t \sigma\mathbf{n}\, dv\,. \tag{A.6}$$

Recalling that $\sigma\mathbf{n} = \mathbf{s}$ on the boundary and using Eqs. (A.5,A.6) in (A.4), we obtain the *virtual work equation* in the deformed configuration as

$$\int_v \nabla\delta\mathbf{u} : \sigma\, dv - \int_v \delta\mathbf{u}^t\mathbf{b}\, dv - \int_{\partial v} \delta\mathbf{u}^t\mathbf{s}\, da = 0\,. \tag{A.7}$$

This expression owes its name to the fact that, when interpreting $\delta\mathbf{u}$ as *virtual displacements*, the last two terms are products of external force and virtual displacement, i.e., external virtual work. Similarly, the first integral term corresponds to internal virtual work.

The integration domain in Eq. (A.7) as well as the Cauchy stress and the gradient operator refer to the deformed configuration. Since the deformed state is unknown, we have to rephrase this expression in terms of a known reference configuration. In the Total Lagrangian finite element formulation [Bat96], which we pursue here, this reference configuration is the rest state, also referred to as the material configuration. In the following, we will focus on the first part of Eq. (A.7) that corresponds to the internal virtual work and start by emphasizing the spatial nature of the gradient operator,

$$\nabla\delta\mathbf{u} = \frac{\partial\delta\mathbf{u}(\mathbf{x})}{\partial\mathbf{x}} = \nabla_t\delta\mathbf{u}\,. \tag{A.8}$$

Next, we observe the relation between material and spatial gradient operators

$$\nabla_0\delta\mathbf{u} = \frac{\partial\delta\mathbf{u}(\bar{\mathbf{x}})}{\partial\bar{\mathbf{x}}} = \frac{\partial\delta\mathbf{u}(\mathbf{x})}{\partial\mathbf{x}}\frac{\partial\mathbf{x}}{\partial\bar{\mathbf{x}}} = \nabla_t\delta\mathbf{u}\mathbf{F}\,. \tag{A.9}$$

Furthermore, we note the fact that an element of current volume dv is related to its undeformed counterpart dV as $dv = JdV$, where $J = \det\mathbf{F}$. We can thus relate the internal virtual work in the deformed state to the material configuration as

$$\int_v \sigma(\mathbf{x}) : \nabla_t\delta\mathbf{u}(\mathbf{x})\, dv = \int_V J\sigma(\bar{\mathbf{x}}) : \nabla_0\delta\mathbf{u}(\bar{\mathbf{x}})\mathbf{F}^{-1}\, dV\,. \tag{A.10}$$

This expression is not yet satisfying since it uses the Cauchy stress σ, which is a spatial quantity. In order to arrive at a pure material formulation, we have to apply further transformations. Using the properties of the trace operator

$$\mathbf{A} : \mathbf{B} = \text{tr}(\mathbf{A}^t\mathbf{B}) = \text{tr}(\mathbf{B}\mathbf{A}^t) = \text{tr}(\mathbf{B}^t\mathbf{A}) = \text{tr}(\mathbf{A}\mathbf{B}^t) \tag{A.11}$$

[1] differential operators will be applied to the test functions

[2] This identity follows from the product rule and can be verified in component form (see Bonet et al. [BW97]).

we transform the integrand on the right hand side of (A.10) as

$$\begin{aligned} J\sigma : \nabla_0 \delta\mathbf{u}\mathbf{F}^{-1} &= J\mathrm{tr}(\sigma\nabla_0\delta\mathbf{u}\mathbf{F}^{-1}) = J\mathrm{tr}(\nabla_0\delta\mathbf{u}\mathbf{F}^{-1}\sigma) \\ &= J\mathrm{tr}(\mathbf{F}^{-1}\sigma\nabla_0\delta\mathbf{u}) = J\sigma\mathbf{F}^{-t} : \nabla_0\delta\mathbf{u} \\ &= \mathbf{P} : \delta\mathbf{F} \end{aligned} \tag{A.12}$$

where $\mathbf{P}$ is the first Piola-Kirchhoff tensor and $\delta\mathbf{F}$ denotes the variation of the deformation gradient. The occurrence of $\mathbf{P}$ is still inconvenient since it is an unsymmetric tensor that maps normal directions in the material configuration to traction forces in the deformed configuration. A pure material formulation is finally obtained by symmetrizing $\mathbf{P}$ as

$$\begin{aligned} \mathbf{P} : \delta\mathbf{F} &= J\mathbf{F}\mathbf{F}^{-1}\sigma\mathbf{F} : \nabla_0\delta\mathbf{u} = J\mathrm{tr}(\mathbf{F}^t\sigma\mathbf{F}^{-t}\mathbf{F}^t\nabla_0\delta\mathbf{u}) \\ &= J\mathbf{F}^t\sigma\mathbf{F}^{-t} : \mathbf{F}^t\nabla_0\delta\mathbf{u} \\ &= \mathbf{S} : \delta\mathbf{E}\,, \end{aligned} \tag{A.13}$$

where $\mathbf{S}$ is the second Piola-Kirchhoff tensor and $\delta\mathbf{E}$ is the variation of the Green strain

$$\delta\mathbf{E} = \frac{1}{2}\mathbf{F}^t\nabla_0\delta\mathbf{u} + \nabla_0\delta\mathbf{u}^t\mathbf{F}\,. \tag{A.14}$$

Note that $\mathbf{S}$ is a symmetric tensor that maps normal directions in the material configuration to traction forces in the material configuration. To summarize, we restate the internal virtual work in the material configuration,

$$\delta W_{int} = \int_V \mathbf{S} : \delta\mathbf{E}\; dV\,, \tag{A.15}$$

which is the basis for the finite element discretization introduced subsequently.

A.2 Finite Element Discretization

A finite element mesh provides a geometric decomposition of the domain Ω into elemental domains $\Omega_e \subset \Omega$ which satisfy $\Omega = \cup_i \Omega_i$. An analogous decomposition holds for the integral of the internal virtual work

$$\delta W_{int} = \int_V \mathbf{S} : \delta\mathbf{E}\; dV = \sum_e \int_{\Omega_e} \mathbf{S} : \delta\mathbf{E}\; dV = \sum_e \delta W^e_{int}\,. \tag{A.16}$$

In order to simplify notation, we will only consider the contribution of a single element in the following. For further convenience, we assemble the current nodal positions of the element into a matrix $\mathbf{x}^e \in \mathbb{R}^{3\times 3}$ and write its nodal shape functions as a vector $\mathbf{N} \in \mathbb{R}^3$. Our goal is to establish discrete expressions for the integral term Eq. (A.16) and we begin by replacing the virtual displacement field with its finite element approximation

$$\delta\mathbf{u} = (\delta\mathbf{u}^e)^t\mathbf{N}\,. \tag{A.17}$$

Recalling the definition of the variation of Green's strain from Eq. (A.14), we have

$$\mathbf{S} : \delta\mathbf{E} = \mathbf{S} : \frac{1}{2}(\mathbf{F}^t\nabla_0\delta\mathbf{u} + \nabla_0\delta\mathbf{u}^t\mathbf{F}) = \mathbf{S} : \mathbf{F}^t\nabla_0\delta\mathbf{u} = \nabla_0\delta\mathbf{u} : \mathbf{FS} \; . \tag{A.18}$$

Inserting (A.17) for $\delta\mathbf{u}$ yields

$$\begin{aligned} \nabla_0\delta\mathbf{u} : \mathbf{FS} &= \mathrm{tr}((\nabla_0\mathbf{N})^t\delta\mathbf{u}^e\mathbf{FS}) = \mathrm{tr}((\delta\mathbf{u}^e)^t\nabla_0\mathbf{NSF}^t) && \text{(A.19)} \\ &= \delta\mathbf{u}^e : \nabla_0\mathbf{NSF}^t \; , && \text{(A.20)} \end{aligned}$$

from which we identify the matrix of nodal force densities as

$$\mathbf{f}^e = \nabla_0\mathbf{NSF}^t \; . \tag{A.21}$$

Finally, discrete nodal forces are obtained by integrating over the element (2.32) and evaluating the integral using numerical quadrature (2.33) as explained in Sec. 2.2.1.

Appendix B

References

B.1 Authored Publications

Journals

[1] B. Thomaszewski, S. Pabst, and W. Straßer. Continuum-based strain limiting. *Computer Graphics Forum (Proc. of Eurographics '09)*, 28:569–576, 2009.

[2] J. Mezger, B. Thomaszewski, S. Pabst, and W. Straßer. Interactive physically-based shape editing. *Comput. Aided Geom. Des.*, 26(6):680–694, 2009.

[3] B. Thomaszewski, A. Gumann, S. Pabst, and W. Straßer. Magnets in motion. *ACM Trans. Graph. (Proceedings of SIGGRAPH Asia '08)*, 27(5):1–9, 2008.

[4] B. Thomaszewski, S. Pabst, and W. Blochinger. Parallel techniques for physically-based simulation on multi-core processor architectures. *Computers & Graphics*, 31(1):25–40, 2008.

[5] B. Thomaszewski and W. Blochinger. Physically-based simulation of cloth on distributed memory architectures. *Parallel Computing*, 33:377–390, 2007.

Conference Proceedings

[6] S. Pabst, B. Thomaszewski, and W. Straßer. Anisotropic friction for deformable surfaces and solids. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '09)*, pages 149–154, 2009.

[7] B. Thomaszewski, S. Pabst, and W. Straßer. Asynchronous cloth simulation. In *Proceedings of Computer Graphics International (CGI '08)*, 2008.

[8] J. Mezger, B. Thomaszewski, S. Pabst, and W. Straßer. Interactive physically-based shape editing. In *Proceedings of the ACM Symposium on Solid and Physical Modeling (SPM '08)*, pages 79–89, 2008.

[9] S. Pabst, S. Krzywinski, A. Schenk, and B. Thomaszewski. Seams and bending in cloth simulation. In *Proceedings of EG Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS '08)*, 2008.

[10] B. Thomaszewski, S. Pabst, and W. Blochinger. Exploiting parallelism in physically-based simulations on multi-core processor architectures. In *Proceedings of Eurographics Symposium on Parallel Graphics and Visualization (EGPGV '07)*, 2007.

[11] B. Thomaszewski, M. Wacker, and W. Straßer. A consistent bending model for cloth simulation with corotational subdivision finite elements. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '06)*.

[12] B. Thomaszewski and W. Blochinger. Parallel simulation of cloth on distributed memory architectures. In *Proceedings of Eurographics Symposium on Parallel Graphics and Visualization (EGPGV '06)*, 2006.

[13] B. Thomaszewski and M. Wacker. Bending models for thin flexible objects. In *WSCG Short Communication Proceedings*, 2006.

[14] M. Keckeisen, S. Kimmerle, B. Thomaszewski, and M. Wacker. Modelling effects of wind fields in cloth animation. In *WSCG*, pages 205–212, 2004.

Tutorials

[15] B. Thomaszewski, M. Wacker, Straßer W., N. Magnenat-Thalmann, and E. Lyard. Advanced topics in virtual garment simulation. In *Proc. of Eurographics '07, Tutorial 10*, 2007.

[16] M. Wacker, W. Straßer, B. Thomaszewski, N. Magnenat-Thalmann, and P. Volino. High performance virtual garment simulation. In *Proc. of Eurographics '06, Tutorial 6*, 2006.

[17] N. Magnenat-Thalmann, P. Volino, M. Wacker, B. Thomaszewski, and M. Keckeisen. Key techniques for interactive virtual garment simulation. In *Proceedings of Eurographics '05, Tutorial 4*, 2005.

Technical Reports

[18] P. Decaudin, B. Thomaszewski, and M.-P. Cani. Virtual garments based on geometric features of fabric buckling. Technical Report 5549, INRIA, 2005.

[19] J. Mezger, B. Thomaszewski, S. Pabst, and W. Straßer. A finite element method for interactive physically-based shape modelling with quadratic tetrahedra. Technical Report WSI-2007-01, Universität Tübingen, 2007.

B.2 Bibliography

[AG85] W. Armstrong and M. Green. The dynamics of articulated rigid bodies for purposes of animation. *The Visual Computer*, 1:231–240, 1985.

[AG00] T. M. Atanackovic and A. Guran. *Theory of Elasticity for Scientists and Engineers*. Birkhäuser, 2000.

[AMHH08] T. Akenine-Möller, E. Haines, and N. Hoffman. *Real-Time Rendering*. A. K. Peters, Ltd., 3rd edition, 2008.

[AP97] M. Anitescu and F. A. Potra. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics*, 14(3):231–247, 1997.

[AP98] U. Ascher and L. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Siam, 1st edition, 1998.

[Arn97] V. I. Arnold. *Mathematical Methods of Classical Mechanics*. Springer, 2nd edition, 1997.

[AY84] G. Akoun and J.-P. Yonnet. 3d analytical calculation of the forces exerted between two cuboidal magnets. *IEEE Trans. Magn.*, 20(5):1962–1964, 1984.

[BA04] E. Boxerman and U. Ascher. Decomposing cloth. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '04)*, pp. 153–161, 2004.

[Bar89a] D. Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *Proceedings of ACM SIGGRAPH '89*, pp. 223–232, 1989.

[Bar91] D. Baraff. Coping with friction for non-penetrating rigid body simulation. In *Proceedings of ACM SIGGRAPH '91*, pp. 31–41, 1991.

[Bar94] D. Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *Proceedings of ACM SIGGRAPH '94*, pp. 23–34, 1994.

[Bar96] D. Baraff. Linear-time dynamics using Lagrange multipliers. In *Proceedings of ACM SIGGRAPH '96*, pp. 137–146, 1996.

[Bar01] D. Baraff. Rigid body simulation. In *Physically Based Modeling: SIGGRAPH '01 Course 25*, 2001.

[BASH00] K. Brown, S. Attaway, S.J.Plimpton, and B. Hendrickson. Parallel strategies for crash and impact simulations. *Computer Methods in Applied Mechanics and Engineering*, 184:375–390, 2000.

[Bat96] K.-J. Bathe. *Finite element procedures*. Prentice Hall, New Jersey, 1996.

[BB88b] R. Barzel and A. H. Barr. A modeling system based on dynamic constraints. In *Proceedings of ACM SIGGRAPH '88*, pp. 179–188, 1988.

[BC00] D. Bourguignon and M.-P. Cani. Controlling anisotropy in mass-spring systems. In *Proceedings of Eurographics Workshop on Computer Animation and Simulation (EGCAS '00)*, pp. 113–123, 2000.

[BFA02] R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. In *Proceedings of ACM SIGGRAPH '02*, pp. 594–603, 2002.

[BH86] J. Barnes and P. Hut. A hierarchical O(N log N) force-calculation algorithm. *Nature*, 324(4):446–449, 1986.

[BHW94] D. Breen, D. House, and M. Wozny. Predicting the drape of woven cloth using interacting particles. In *Proceedings of ACM SIGGRAPH '94*, pp. 365–372, 1994.

[BKLW99] W. Blochinger, W. Küchlin, C. Ludwig, and A. Weber. An object-oriented platform for distributed high-performance Symbolic Computation. *Mathematics and Computers in Simulation*, 49:161–178, 1999.

[BMF03] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '03)*, pp. 28–36, 2003.

[Box03] E. Boxerman. Speeding up cloth simulation. Master's thesis, The University of British Columbia, 2003.

[BW97] J. Bonet and R. D. Wood. *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, 1997.

[BW98] D. Baraff and A. Witkin. Large steps in cloth simulation. In *Proceedings of ACM SIGGRAPH '98*, pp. 43–54, 1998.

[BW04] G. Baciu and W. S.-K. Wong. Image-based collision detection for deformable cloth models. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):649–663, 2004.

[BWH+06] M. Bergou, M. Wardetzky, D. Harmon, D. Zorin, and E. Grinspun. A quadratic bending model for inextensible surfaces. In *Fourth Eurographics Symposium on Geometry Processing (SGP '06)*, pp. 227–230, 2006.

[BWK03] D. Baraff, A. Witkin, and M. Kass. Untangling cloth. In *Proceeding of ACM SIGGRAPH '03*, pp. 862–870, 2003.

[CCOS91] J. R. Collier, B. J. Collier, G. O'Toole, and S. M. Sargand. Drape prediction by means of finite-element analysis. *J. Text. Inst.*, 82(1):96–107, 1991.

[CF00] S. Chenney and D. A. Forsyth. Sampling plausible solutions to multibody constraint problems. In *Proceedings of ACM SIGGRAPH '00*, pp. 219–228, 2000.

[CK02] K.-J. Choi and H.-S. Ko. Stable but responsive cloth. In *Proceedings of ACM SIGGRAPH '02*, pp. 604–611, 2002.

[CMT04] M. Carlson, P. J. Mucha, and G. Turk. Rigid fluid: animating the interplay between rigid bodies and fluid. In *Proceedings of ACM SIGGRAPH '04*, pp. 377–384, 2004.

[CO01] F. Cirak and M. Ortiz. Fully C^1-conforming subdivision elements for finite deformation thin-shell analysis. *Journal for Numerical Methods in Engineering*, 51:813–833, 2001.

[Coh92] M. F. Cohen. Interactive spacetime control for animation. In *Proceedings of ACM SIGGRAPH '92*, pp. 293–302, 1992.

[COS00] F. Cirak, M. Ortiz, and P. Schröder. Subdivision surfaces: A new paradigm for thin-shell finite-element analysis. *Journal for Numerical Methods in Engineering*, 47:2039–2072, 2000.

[Cot77] R. W. Cottle. Numerical methods for complementarity problems in engineering and applied science. In *Computing Methods in Applied Sciences and Engineering*, pp. 37–52. Springer, 1977.

[CTM08] S. Curtis, R. Tamstorf, and D. Manocha. Fast collision detection for deformable models using representative triangles. In *Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games (I3D '08)*, pp. 61–69, 2008.

[CW05] F. Cirak and M. West. Decomposition-based contact response (DCR) for explicit finite element dynamics. *International Journal for Numerical Methods in Engineering*, 64:1078–1110, 2005.

[CYTT92] M. Carignan, Y. Yang, N. M. Thalmann, and D. Thalmann. Dressing animated synthetic actors with complex deformable clothes. In *Proceedings of ACM SIGGRAPH '92*, pp. 99–104, 1992.

[DDCB01] G. Debunne, M. Desbrun, M.-P. Cani, and A. H. Barr. Dynamic real-time deformations using space & time adaptive sampling. In *Proceedings of ACM SIGGRAPH '03*, pp. 31–36, 2001.

[Del01] F. Delfino. Some numerical aspects in electrodynamics of magnetic materials. *ICS Newsletter*, 8(3), 2001.

[Del08] H. Delingette. Triangular springs for modeling nonlinear membranes. *IEEE Transactions on Visualization and Computer Graphics*, 14(2):329–341, 2008.

[DMMR01] F. Delfino, A. Manella, P. Molfino, and M. Rossi. Numerical calculation of total force upon permanent magnets using equivalent source methods. *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, 20(2):431–447, 2001.

[dMRM98] L. H. de Medeiros, G. Reyne, and G. Meunier. Comparison of global force calculations on permanent magnets. *IEEE Trans. Magn.*, 34(5):3560–3563, 1998.

[EB00] J. Eischen and R. Bigliani. Continuum versus particle representations. In D. House and D. Breen (Editors), *Cloth Modeling and Animation*, pp. 79–122. 2000.

[EB08] E. English and R. Bridson. Animating developable surfaces using nonconforming elements. In *Proceedings of ACM SIGGRAPH '08*, 2008.

[EDC96] J. Eischen, S. Deng, and T. Clapp. Finite-element modeling and control of flexible fabric parts. *IEEE Computer Graphics and Applications*, 16(5):71–80, 1996.

[EEH00] B. Eberhardt, O. Etzmuß, and M. Hauth. Implicit-explicit schemes for fast animation with particle systems. In *Eurographics Computer Animation and Simulation Workshop '00*, 2000.

[EGS03] O. Etzmuß, J. Gross, and W. Straßer. Deriving a particle system from continuum mechanics for the animation of deformable objects. *IEEE Transactions on Visualization and Computer Graphics*, 9(4):538–550, 2003.

[EKS03] O. Etzmuß, M. Keckeisen, and W. Straßer. A Fast Finite Element Solution for Cloth Modelling. In *Proceedings of Pacific Graphics (PG '03)*, pp. 244–251, 2003.

[EMS00] B. Eberhardt, M. Meissner, and W. Straßer. Knit fabrics. In *Cloth modeling and animation*, pp. 123–144. A. K. Peters, Ltd., 2000.

[Erl07] K. Erleben. Velocity-based shock propagation for multibody dynamics animation. *ACM Trans. Graph.*, 26(2):12, 2007.

[ES08] J. T. E. Sifakis, S. Marino. Globally coupled impulse-based collision handling for cloth simulation. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '08)*, 2008.

[EWS96] B. Eberhardt, A. Weber, and W. Straßer. A fast, flexible, particle-system model for cloth draping. *IEEE Computer Graphics and Applications*, 16(5):52–59, 1996.

[FBAF08] F. Faure, S. Barbier, J. Allard, and F. Falipou. Image-based collision detection and response between arbitrary volume objects. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '08)*, pp. 155–162, 2008.

[Fea87] R. Featherstone. *Robot Dynamics Algorithms*. Kluwer Academic Publishers, 1987.

[FPRJ00] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *Proceedings of ACM SIGGRAPH '00*, pp. 249–254, 2000.

[Fun93] Y. C. Fung. *Biomechanics: Mechanical properties of living tissues*. Springer, 1993.

[GBF03] E. Guendelman, R. Bridson, and R. Fedkiw. Nonconvex rigid bodies with stacking. In *Proceedings of ACM SIGGRAPH '03*, pp. 871–878, 2003.

[GGWZ07] A. Garg, E. Grinspun, M. Wardetzky, and D. Zorin. Cubic shells. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '07)*, pp. 91–98, 2007.

[GHDS03] E. Grinspun, A. Hirani, M. Desbrun, and P. Schröder. Discrete shells. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '03)*, pp. 62–67, 2003.

[GHF+07] R. Goldenthal, D. Harmon, R. Fattal, M. Bercovier, and E. Grinspun. Efficient simulation of inextensible cloth. In *Proceedings of ACM SIGGRAPH '07*, pp. 281–290, 2007.

[GJK88] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, 4(2):193–203, 1988.

[GKJ+05] N. Govindaraju, D. Knott, N. Jain, I. Kabul, R. Tamstorf, R. Gayle, M. Lin, and D. Manocha. Interactive collision detection between deformable models using chromatic decomposition. In *Proceedings of ACM SIGGRAPH '05*, 2005.

[GKS02] E. Grinspun, P. Krysl, and P. Schröder. Charms: a simple framework for adaptive simulation. In *Proceedings of ACM SIGGRAPH '02*, pp. 281–290, 2002.

[GLM96] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: a hierarchical structure for rapid interference detection. In *Proceedings of ACM SIGGRAPH '96*, pp. 171–180, 1996.

[GLS95] L. Gan, N. Ly, and G. Steven. A finite element analysis of the draping of fabric. *Textile Research Journal*, 65(11):660–668, 1995.

[GRR+05] E. Gutierréz, S. Romero, L. F. Romero, O. Plata, and E. L. Zapata. Parallel techniques in irregular codes: cloth simulation as case of study. *Journal of Parallel and Distributed Computing*, 65(4):424–436, 2005.

[GRV95] T. Gautier, J. Roch, and G. Villard. Regular versus irregular problems and algorithms. In *In Proceedings of IRREGULAR '95*. Springer, 1995.

[GVL96] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

[Hah88] J. K. Hahn. Realistic animation of rigid bodies. In *Proceedings of ACM SIGGRAPH '88*, pp. 299–308, 1988.

[Hau04] M. Hauth. Visual simulation of deformable models. Phd thesis, Wilhelm-Schickard-Institut für Informatik, University of Tübingen, Germany, 2004.

[HB00] D. H. House and D. E. Breen (Editors). *Cloth modeling and animation*. A. K. Peters, 2000.

[HCJ+05] M. Hong, M.-H. Choi, S. Jung, S. Welch, and J. Trapp. Effective constrained dynamic simulation using implicit constraint enforcement. In *Proceedings of International Conference on Robotics and Automation*, pp. 4520–4525, 2005.

[HE01a] M. Hauth and O. Etzmuß. A High Performance Solver for the Animation of Deformable Objects using Advanced Numerical Methods. In *Computer Graphics Forum (Proceedings of Eurographics '01)*, pp. 319–328, 2001.

[HES03] M. Hauth, O. Etzmuß, and W. Straßer. Analysis of numerical methods for the simulation of deformable models. *The Visual Computer*, 19:586–600, 2003.

[HF07] M. Hutter and A. Fuhrmann. Optimized continuous collision detection for deformable triangle meshes. In *Journal of WSCG*, pp. 25–32, 2007.

[HGS03] M. Hauth, J. Groß, and W. Straßer. Interactive physically based solid dynamics. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '03)*, pp. 17–27, 2003.

[HLW03] E. Hairer, C. Lubich, and G. Wanner. Geometric numerical integration illustrated by the störmer–verlet method. *Acta Numerica*, 12:399–450, 2003.

[HLW06] E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration*. Springer, Berlin, 2nd edition, 2006.

[HS04] M. Hauth and W. Straßer. Corotational simulation of deformable solids. *Journal of WSCG*, 12(1):137–145, 2004.

[HTG04] B. Heidelberger, M. Teschner, and M. Gross. Detection of collisions and self-collisions using image-space techniques. In *Journal of WSCG*, pp. 145–152, 2004.

[Hub96] P. M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Trans. Graph.*, 15(3):179–210, 1996.

[HVS+09] D. Harmon, E. Vouga, B. Smith, R. Tamstorf, and E. Grinspun. Asynchronous contact mechanics. In *Proceedings of ACM SIGGRAPH '09*, 2009.

[HVTG08] D. Harmon, E. Vouga, R. Tamstorf, and E. Grinspun. Robust treatment of simultaneous collisions. In *Proceedings of ACM SIGGRAPH '08*, 2008.

[HW02] E. Hairer, , and G. Wanner. *Solving Ordinary Differential Equations 2: Stiff and Differential-Algebraic Problems*. Springer, Berlin, 2nd edition, 2002.

[HWN08] E. Hairer, G. Wanner, and S. P. Noersett. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer, Berlin, 3rd edition, 2008.

[IA04] M. Itskov and N. Aksel. A constitutive model for othotropic elastoplasticity at large strains. *Archive of Applied Mechanics*, 74:75–91, 2004.

[IC87] P. M. Isaacs and M. F. Cohen. Controlling dynamic simulation with kinematic constraints. In *Proceedings of ACM SIGGRAPH '87*, pp. 215–224, 1987.

[Jac99] J. D. Jackson. *Classical Electrodynamics*. Wiley, New York, 3rd edition, 1999.

[JP04] D. L. James and D. K. Pai. BD-Tree: Output-sensitive collision detection for reduced deformable models. In *Proceedings of ACM SIGGRAPH '04*, 2004.

[Kar03] G. Karypis. Multi-constraint mesh partitioning for contact/impact computations. In *Proceedings of the 2003 ACM/IEEE Conference on Supercomputing*, 2003.

[Kaw80a] S. Kawabata. *The Standardization and Analysis of Hand Evaluation*. The Textile Machinery Society of Japan, Osaka, 1980.

[KB04] M. Keckeisen and W. Blochinger. Parallel implicit integration for cloth animations on distributed memory architectures. In *Proceedings of Eurographics Symposium on Parallel Graphics and Visualization (EGPGV '04)*, 2004.

[KE04] T. Klein and T. Ertl. Illustrating magnetic field lines using a discrete particle model. In *Proceedings of the Workshop on Vision, Modelling, and Visualization 2004 (VMV '04)*, pp. 387–394, 2004.

[KEP05] D. M. Kaufman, T. Edmunds, and D. K. Pai. Fast frictional dynamics for rigid bodies. In *Proceedings of ACM SIGGRAPH '05*, pp. 946–956, 2005.

[KFCO06] B. M. Klingner, B. E. Feldman, N. Chentanez, and J. F. O'Brien. Fluid animation with dynamic meshes. In *Proceedings of ACM SIGGRAPH '06*, 2006.

[KHM+98] J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of *k*-DOPs. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998.

[KJM08] J. M. Kaldor, D. L. James, and S. Marschner. Simulating knitted cloth at the yarn level. In *Proceedings of ACM SIGGRAPH '08*, 2008.

[KK96b] G. Karypis and V. Kumar. Parallel Multilevel k-way Partitioning Schemes for Irregular Graphs. Technical Report 036, University of Minneapolis, 1996.

[KL04] T. Kim and M. C. Lin. Physically based animation and rendering of lightning. In *Proceedings of the Computer Graphics and Applications (PG '04)*, pp. 267–275, 2004.

[KMB+09] P. Kaufmann, S. Martin, M. Botsch, E. Grinspun, and M. Gross. Enrichment textures for detailed cutting of shells. In *Proceedings of ACM SIGGRAPH '09*, 2009.

[KMBG09] P. Kaufmann, S. Martin, M. Botsch, and M. Gross. Flexible simulation of deformable models using discontinuous Galerkin FEM. *Graph. Models*, 71(4):153–167, 2009.

[KNF04] S. Kimmerle, M. Nesme, and F. Faure. Hierarchy accelerated stochastic collision detection. In *9th International Workshop on Vision, Modeling, and Visualization (VMV '04)*, pp. 307–314, 2004.

[KSJP08] D. M. Kaufman, S. Sueda, D. L. James, and D. K. Pai. Staggered projections for frictional contact in multibody systems. *ACM Trans. Graph. (Proceedings SIGGRAPH Asia '08)*, 27(5):1–11, 2008.

[KTY09] R. Kikuuwe, H. Tabuchi, and M. Yamamoto. An edge-based computationally efficient formulation of Saint Venant-Kirchhoff tetrahedral finite elements. *ACM Trans. Graph.*, 28(1):1–13, 2009.

[KYT+06] L. Kharevych, W. Yang, Y. Tong, E. Kanso, J. E. Marsden, P. Schröder, and M. Desbrun. Geometric, variational integrators for computer animation. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '06)*, pp. 43–51, 2006.

[KZ03] J. Klein and G. Zachmann. ADB-Trees: Controlling the error of time-critical collision detection. In *8th International Workshop on Vision, Modeling, and Visualization, (VMV '03)*, pp. 37–46, 2003.

[LAM01] T. Larsson and T. Akenine-Möller. Collision detection for continuously deforming bodies. *Proceedings of Eurographics '01*, pp. 325–333, 2001.

[LC91] M. C. Lin and J. F. Canny. A fast algorithm for incremental distance calculation. In *IEEE International Conference on Robotics and Automation*, pp. 1008–1014, 1991.

[LC98] T.-Y. Li and J.-S. Chen. Incremental 3d collision detection with hierarchical data structures. In *VRST '98: Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, pp. 139–144, 1998.

[Lew03] A. Lew. *Variational Time Integrators in Computational Solid Mechanics*. Phd thesis, California Institute of Technology, 2003.

[LG98] M. C. Lin and S. Gottschalk. Collision detection between geometric models: A survey. In *IMA Conference on Mathematics of Surfaces*, 1998.

[LGPT01] R. Lario, C. Garcia, M. Prieto, and F. Tirado. Rapid parallelization of a multilevel cloth simulator using OpenMP. In *Third European Workshop on OpenMP*, 2001.

[LH06] S. Lefebvre and H. Hoppe. Perfect spatial hashing. In *Proceedings of ACM SIGGRAPH '06*, pp. 579–588, 2006.

[LK02] O. S. Lawlor and L. V. Kalée. A voxel-based parallel collision detection algorithm. In *ICS '02: Proceedings of the 16th International Conference on Supercomputing*, pp. 285–293. ACM, 2002.

[Llo80] D. Lloyd. The analysis of complex fabric deformations. In *Mechanics of Flexible Fiber Assemblies*, pp. 311–342, 1980.

[LLP84] L. D. Landau, E. M. Lifshitz, and L. P. Pitaevskii. *Electrodynamics of Continuous Media*. Pergamon Press, Oxford, 2nd edition, 1984.

[LMOW03] A. Lew, J. E. Marsden, M. Ortiz, and M. West. Asynchronous variational integrators. *Archive for Rational Mechanics and Analysis*, 167:85–146, 2003.

[Löt84] P. Lötstedt. Numerical simulation of time-dependent contact and friction problems in rigid body mechanics. *SIAM Journal on Scientific and Statistical Computing*, 5(2):370–393, 1984.

[LSH07] B. Lloyd, G. Székely, and M. Harders. Identification of spring parameters for deformable object simulation. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):1081–1094, 2007.

[LV05] L. Li and V. Volkov. Cloth animation with adaptively refined meshes. In *ACSC '05: Proceedings of the Twenty-Eighth Australasian Conference on Computer Science*, pp. 107–113. Australian Computer Society, 2005.

[MC95] B. Mirtich and J. Canny. Impulse-based simulation of rigid bodies. In *SI3D '95: Proceedings of the Symposium on Interactive 3D Graphics*. ACM, 1995.

[MDM+02] M. Müller, J. Dorsey, L. McMillan, R. Jagnow, and B. Cutler. Stable real-time deformations. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '02)*, pp. 49–54, 2002.

[MDSB03a] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*, pp. 35–57. 2003.

[Mez08] J. Mezger. Simulation and animation of deformable bodies. Phd thesis, Wilhelm-Schickard-Institut für Informatik, University of Tübingen, Germany, 2008.

[MHHR06] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position based dynamics. In *Virtual Reality Interactions and Physical Simulation (VRIPHYS)*, pp. 71–80, 2006.

[Mil96] V. J. Milenkovic. Position-based physics: simulating the motion of many highly interacting spheres and polyhedra. In *Proceedings of ACM SIGGRAPH '96*, pp. 129–136, 1996.

[Mir98] B. Mirtich. V-clip: Fast and robust polyhedral collision detection. *ACM Trans. Graph.*, 17:177–208, 1998.

[Mir00] B. Mirtich. Timewarp rigid body simulation. In *Proceedings of ACM SIGGRAPH '00*, pp. 193–200, 2000.

[MKE03] J. Mezger, S. Kimmerle, and O. Etzmuß. Hierarchical techniques in collision detection for cloth animation. *Journal of WSCG*, 11(2):322–329, 2003.

[Möl97] T. Möller. A fast triangle-triangle intersection test. *J. Graph. Tools*, 2(2):25–30, 1997.

[MS01] V. J. Milenkovic and H. Schmidl. Optimization-based animation. In *Proceedings of ACM SIGGRAPH '01*, pp. 37–46, 2001.

[MSP88] C. Magele, H. Stogner, and K. Preis. Comparison of different finite element formulations for 3d magnetostatic problems. *IEEE Transactions on Magnetics*, 24(1):31–34, 1988.

[MTPS08] J. Mezger, B. Thomaszewski, S. Pabst, and W. Straßer. Interactive physically-based shape editing. In *SPM '08: Proceedings of the 2008 ACM symposium on Solid and Physical Modeling*, pp. 79–89, 2008.

[Mül08] M. Müller. Hierarchical position based dynamics. In *Virtual Reality Interactions and Physical Simulation (VRIPHYS)*, pp. 1–10, 2008.

[MW88] M. Moore and J. Wilhelms. Collision detection and response for computer animationr. In *Proceedings of ACM SIGGRAPH '88*, pp. 289–298, 1988.

[MW01] J. E. Marsden and M. West. Discrete mechanics and variational integrators. *Acta Numerica*, 10:357–514, 2001.

[NS74] O. Nevanlinna and A. Sippilä. A nonexistence theorem for A-stable explicit methods. *Mathematics Of Computation*, 28(128):1053–1055, 1974.

[OCSG07] M. A. Otaduy, O. Chassot, D. Steinemann, and M. Gross. Balanced hierarchies for collision detection between fracturing objects. In *IEEE Virtual Reality Conference*, pp. 83–90, 2007.

[OF02] S. J. Osher and R. P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 1st edition, 2002.

[OH99] J. F. O'Brien and J. K. Hodgins. Graphical modeling and animation of brittle fracture. In *Proceedings of ACM SIGGRAPH '99*, pp. 137–146, 1999.

[OTSG09] M. A. Otaduy, R. Tamstorf, D. Steinemann, and M. Gross. Implicit contact handling for deformable objects. *Computer Graphics Forum (Proceedings of Eurographics '09)*, 28(2), 2009.

[OZH00] J. F. O'Brien, V. B. Zordan, and J. K. Hodgins. Combining active and passive simulations for secondary motion. *IEEE Comput. Graph. Appl.*, 20(4):86–96, 2000.

[PAH+98] S. Plimpton, S. Attaway, B. Hendrickson, J. Swegle, and C. Vanghan. Parallel transient dynamics simulations. *J. Parallel Distrib. Comput.*, 50(1-2):104–122, 1998.

[PB81] S. M. Platt and N. I. Badler. Animating facial expressions. In *Proceedings of ACM SIGGRAPH '81*, pp. 245–252, 1981.

[PG95] I. J. Palmer and R. L. Grimsdale. Collision detection for animation using sphere-trees. *Computer Graphics Forum*, 14(2):105–116, 1995.

[PKST08] S. Pabst, S. Krzywinski, A. Schenk, and B. Thomaszewski. Seams and bending in cloth simulation. In *Virtual Reality Interactions and Physical Simulation (VRIPHYS)*, 2008.

[Pop10] V. L. Popov. *Contact Mechanics and Friction*. Springer, 1st edition, 2010.

[Pro95] X. Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface '95*, pp. 147–154, 1995.

[Pro97] X. Provot. Collision and self-collision handling in cloth model dedicated to design garments. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation (CAS 1997)*, pp. 177–189, 1997.

[PSE+00] J. Popović, S. M. Seitz, M. Erdmann, Z. Popović, and A. Witkin. Interactive manipulation of rigid body simulations. In *Proceedings of ACM SIGGRAPH '00*, pp. 209–217, 2000.

[PSE03] J. Popović, S. M. Seitz, and M. Erdmann. Motion sketching for control of rigid-body simulations. *ACM Trans. Graph.*, 22(4):1034–1054, 2003.

[RH91] M. H. Raibert and J. K. Hodgins. Animation of dynamic legged locomotion. In *Proceedings of ACM SIGGRAPH '91*, pp. 349–358, 1991.

[RK87] V. N. Rao and V. Kumar. Parallel depth first search, part I: Implementation. *International Journal of Parallel Programming*, 16(6):479–499, 1987.

[RMSG+08] A. Robinson-Mosher, T. Shinar, J. Gretarsson, J. Su, and R. Fedkiw. Two-way coupling of fluids to rigid and deformable solids and shells. In *Proceedings of ACM SIGGRAPH '08*, 2008.

[RRZ00] S. Romero, L. F. Romero, and E. L. Zapata. Fast cloth simulation with parallel computers. In *Euro-Par*, pp. 491–499, 2000.

[SF89] J. C. Simo and D. D. Fox. On stress resultant geometrically exact shell model. part I: formulation and optimal parametrization. *Comput. Methods Appl. Mech. Eng.*, 72(3):267–304, 1989.

[SGG+06] A. Sud, N. Govindaraju, R. Gayle, I. Kabul, and D. Manocha. Fast proximity computation among deformable models using discrete Voronoi diagrams. In *Proceedings of ACM SIGGRAPH '06*, pp. 1144–1153, 2006.

[Smi06] R. Smith. Open dynamics engine (ODE), 2006. `http://www.ode.org`.

[SSF08] T. Shinar, C. Schroeder, and R. Fedkiw. Two-way coupling of rigid and deformable bodies. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '08)*, pp. 95–103, 2008.

[SSIF07] E. Sifakis, T. Shinar, G. Irving, and R. Fedkiw. Hybrid simulation of deformable solids. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '07)*, pp. 81–90, 2007.

[SSIF08] A. Selle, J. Su, G. Irving, and R. Fedkiw. Robust high-resolution cloth using parallelism, history-based collisions and accurate friction. *IEEE Transactions on Visualization and Graphics*, 15(2):339–350, 2008.

[ST96] D. Stewart and J. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal of Numerical Methods in Engineering*, 39:2673–2691, 1996.

[Ste00] D. Stewart. Rigid-body dynamics with friction and impact. *SIAM Review*, 42(1):3–39, 2000.

[TB06a] B. Thomaszewski and W. Blochinger. Parallel simulation of cloth on distributed memory architectures. In *Eurographics Symposium on Parallel Graphics and Visualization (EGPGV '06)*, 2006.

[TB07] B. Thomaszewski and W. Blochinger. Physically-based simulation of cloth on distributed memory architectures. *Parallel Computing*, 33:377–390, 2007.

[TCYM09] M. Tang, S. Curtis, S.-E. Yoon, and D. Manocha. ICCD: Interactive continuous collision detection between deformable models using connectivity-based culling. *IEEE Transactions on Visualization and Computer Graphics*, 15(4):544–557, 2009.

[TGPS08] B. Thomaszewski, A. Gumann, S. Pabst, and W. Straßer. Magnets in motion. *ACM Trans. Graph. (Proceedings of SIGGRAPH Asia '08)*, 27(5):1–9, 2008.

[THM+03] M. Teschner, B. Heidelberger, M. Müller, D. Pomerantes, and M. H. Gross. Optimized spatial hashing for collision detection of deformable objects. In *8th International Workshop on Vision, Modeling, and Visualization (VMV '03)*, pp. 47–54, 2003.

[TJ08] C. D. Twigg and D. L. James. Backward steps in rigid body simulation. In *Proceedings of ACM SIGGRAPH '08*, pp. 1–10, 2008.

[TKZ+04] M. Teschner, S. Kimmerle, G. Zachmann, B. Heidelberger, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, and W. Straßer. Collision detection for deformable objects. In *Eurographics State-of-the-Art Report (EG-STAR)*, pp. 119–139, 2004.

[TMT09] M. Tang, D. Manocha, and R. Tong. Multi-core collision detection between deformable models. In *SPM '09: 2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling*, pp. 355–360, 2009.

[TPB07] B. Thomaszewski, S. Pabst, and W. Blochinger. Exploiting parallelism in physically-based simulations on multi-core processor architectures. In *Eurographics Symposium on Parallel Graphics and Visualization (EGPGV '07)*, 2007.

[TPB08] B. Thomaszewski, S. Pabst, and W. Blochinger. Parallel techniques for physically-based simulation on multi-core processor architectures. *Computers & Graphics*, 31(1):25–40, 2008.

[TPBF87] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *Proceedings of ACM SIGGRAPH '87*, pp. 205–214, 1987.

[TPS08] B. Thomaszewski, S. Pabst, and W. Straßer. Asynchronous cloth simulation. In *Proceedings of Computer Graphics International (CGI '08)*, 2008.

[TPS09] B. Thomaszewski, S. Pabst, and W. Straßer. Continuum-based strain limiting. *Computer Graphics Forum (Proceedings of Eurographics '09)*, 28:569–576, 2009.

[Tsi06] K. D. Tsiknis. Better cloth through unbiased strain limiting and physics-aware subdivision. Master's thesis, The University of British Columbia, 2006.

[Tur89] G. Turk. Interactive collision detection for molecular graphics. Master's thesis, University of North Carolina at Chapel Hill, 1989.

[TWS06] B. Thomaszewski, M. Wacker, and W. Straßer. A consistent bending model for cloth simulation with corotational subdivision finite elements. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '06)*, pp. 107–116, 2006.

[TWW+07] B. Thomaszewski, M. Wacker, S. W., N. Magnenat-Thalmann, and L. E. Advanced Topics in Virtual Garment Simulation. In *Proceedings of Eurographics '07, Tutorial 10*, 2007.

[VCMT95] P. Volino, M. Courchesne, and N. Magnenat-Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. In *Proceedings of ACM SIGGRAPH '95*, pp. 137–144, 1995.

[vdB98] G. van den Bergen. Efficient collision detection of complex deformable models using AABB trees. *J. Graph. Tools*, 1998.

[VG98] A. Van Gelder. Approximate simulation of elastic membranes by triangulated spring meshes. *J. Graph. Tools*, 3(2):21–42, 1998.

[VHBZ90] B. Von Herzen, A. H. Barr, and H. R. Zatz. Geometric collisions for time-dependent parametric surfaces. In *Proceedings of ACM SIGGRAPH '90*, pp. 39–48, 1990.

[VL03] V. Volkov and L. Li. Real-time refinement and simplification of adaptive triangular meshes. In *VIS '03: Proceedings of the 14th IEEE Visualization*, 2003.

[VMT94] P. Volino and N. Magnenat-Thalmann. Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. In *Computer Graphics Forum (Proceedings of Eurographics '94)*, pp. 155–166, 1994.

[VMT01] P. Volino and N. Magnenat-Thalmann. Comparing efficiency of integration methods for cloth simulation. In *Proceedings of Computer Graphics International (CGI '01)*, pp. 265–274, 2001.

[VMT05a] P. Volino and N. Magnenat-Thalmann. Accurate garment prototyping and simulation. *Computer-Aided Design & Applications*, 2(5):645–654, 2005.

[VMT05b] P. Volino and N. Magnenat-Thalmann. Implicit midpoint integration and adaptive damping for efficient cloth simulation. *Computer Animation in Virtual Worlds*, 16(3-4):163–175, 2005.

[VMT06a] P. Volino and N. Magnenat-Thalmann. Resolving surface collisions through intersection contour minimization. In *Proceedings of ACM SIGGRAPH '06*, pp. 1154–1159, 2006.

[VMT06b] P. Volino and N. Magnenat-Thalmann. Simple linear bending stiffness in particle systems. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '06)*, pp. 101–105, 2006.

[VMTF09] P. Volino, N. Magnenat-Thalmann, and F. Faure. A simple approach to nonlinear tensile stiffness for accurate cloth simulation. *ACM Trans. Graph.*, 28(4), 2009.

[VSC01] T. Vassilev, B. Spanlang, and Y. Chrysanthou. Fast cloth animation on walking avatars. In *Proceedings of Eurographics '01*, pp. 260–267, 2001.

[WB06] W. S.-K. Wong and G. Baciu. A randomized marking scheme for continuous collision detection in simulation of deformable surfaces. In *VRCIA '06: Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, pp. 181–188, 2006.

[WDGT01] X. Wu, M. S. Downes, T. Goktekin, and F. Tendick. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. In *Proceedings of Eurographics '01*, pp. 349–358. 2001.

[Wei86] J. Weil. The synthesis of cloth objects. In *Proceedings of ACM SIGGRAPH '86*, pp. 49–54, 1986.

[WK88] A. Witkin and M. Kass. Spacetime constraints. In *Proceedings of ACM SIGGRAPH '88*, pp. 159–168, 1988.

[WKK+04] M. Wacker, M. Keckeisen, S. Kimmerle, W. Straßer, V. Luckas, C. Groß, A. Fuhrmann, R. Sarlette, M. Sattler, and R. Klein. Virtual Try-On: Virtuelle Textilien in der Graphischen Datenverarbeitung. *Informatik Spektrum*, 27(6):504–511, 2004.

[WSG05] M. Wicke, D. Steinemann, and M. Gross. Efficient animation of point-sampled thin shells. *Computer Graphics Forum (Proc. of Eurographics '05)*, 24(3):667–676, 2005.

[WT03] G. Wempner and D. Talaslidis. *Mechanics of Solids and Shells: Theories and Approximations*. CRC Press, 2003.

[WTF06] R. Weinstein, J. Teran, and R. Fedkiw. Dynamic simulation of articulated rigid bodies with contact and collision. *IEEE Transactions on Visualization and Computer Graphics*, 12(3):365–374, 2006.

[YOH00] G. D. Yngve, J. F. O'Brien, and J. K. Hodgins. Animating explosions. In *Proceedings of ACM SIGGRAPH '00*, pp. 29–36, 2000.

[ZFV02] F. Zara, F. Faure, and J.-M. Vincent. Physical cloth animation on a PC cluster. In *Proceedings of Eurographics Symposium on Parallel Graphics and Visualization (EGPGV '02)*, 2002.

[ZFV04] F. Zara, F. Faure, and J.-M. Vincent. Parallel simulation of large dynamic system on a PCs cluster: Application to cloth simulation. *International Journal of Computers and Applications*, 2004.

[ZT00a] O. Zienkiewicz and R. Taylor. *The Finite Element Method. Volume 1: The Basis*. Butterworth Heinemann, 5th edition, 2000.

[ZW06] G. Zachmann and R. Weller. Kinetic bounding volume hierarchies for deformable objects. In *ACM International Conference on Virtual Reality Continuum and Its Applications (VRCIA)*, 2006.

[ZY00] D. Zhang and M. Yuen. Collision detection for clothed human animation. In *Proceedings of Pacific Graphics (PG '00)*, 2000.