Improvements to Physically Based Cloth Simulation

A thesis submitted for the degree of
Doctor of Philosophy

Tsz Ho Wong
School of Computer Science and Information Technology,
Science, Engineering, and Technology Portfolio,
RMIT University,
Melbourne, Victoria, Australia.

20th March, 2014

Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; and, any editorial work, paid or unpaid, carried out by a third party is acknowledged.

Tsz Ho Wong
School of Computer Science and Information Technology
RMIT University
20th March, 2014

Acknowledgments

First and foremost, I would like to thank my beloved family members for their tremendous love and encouragement throughout my doctoral studies. My wife, Melody Geng, has been always at hand to proof read my papers and thesis, tolerated and comforted me with great patience when the difficulties of my research made me upset and irritable. My grandma Guixi Dai who raised me up with all her consideration and tenderness, kept supporting and believing in me. My father Luya Wang and my uncle Meichuan Wang inspired me in the field of computer science from the beginning and shared with me all his precious experiences both in research and life. My aunt Jakie Cheong and cousin David Tong have been caring for me for all these years, helping me quickly adapt to the life in Melbourne. My mother Zhuohui Zhang and my sister Caroline Hong, even far away in USA, have been encouraging me all the time during periods of self-doubt. I also would like to thank my parents-in-law Xingxing and Xiaoming Xiao for always having confidence and faith in me.

I am specially indebted to my academic supervisors at RMIT university Geoff Leach and Dr. Fabio Zambetta of the School of Computer Science and Information Technology, who have supported me throughout my doctoral studies with their trust, patience, profound knowledge of computer science and mathematics, as well as sincere and solid advice. Without their help at each stage, this thesis could never have been finished.

I would also like to thank fellow research students Pyarllel Knowles and Adil Alharthi for the many insightful and valuable discussions and advice.

I am furthermore grateful to my best friend Dan Yin and my cousin as well as my best friend Shengchun Wang for sparing time and energy to help me figure out several important questions in my research.

I also owe my thanks to David Harmon for sharing the *reef knot* model. The *cloth-on-ball* and *N-body* benchmarks are courtesy of the UNC Dynamic Scene Benchmarks collection. The *flamenco* benchmark is courtesy of Walt Disney Animation Studios and was provided by Rasmus Tamstorf.

Other friends have helped me in ways too various to specify. I must content myself with naming Alex Qin, Benjamin Thomas, Dilan Wickramaratne, Lin Peng, Mark Zhang, Milton Chen, Xinyu Cao, Vivian Wu and Yang Liu ... the list always rouses in me memories of the most pleasant hours sharing delicious food and interesting life experiences together, making my research days lots of fun and joy.

Finally, I would thank RMIT for offering me scholarships during these four years.

Credits

Portions of the material in this thesis have previously appeared in the following publications:

- Virtual subdivision for GPU based collision detection of deformable objects using a uniform grid, Tsz Ho Wong, Geoff Leach, Fabio Zambetta, The Visual Computer 28 (6-8), 829-838, 2012
- Modelling Bending Behaviour in Cloth Simulation Using Hysteresis, Tsz Ho Wong, Geoff Leach, Fabio Zambetta, Computer Graphics Forum, Vol. 32, Issue 8, 183-194, 2013
- An Improved Friction Model for Cloth Simulation, Tsz Ho Wong, Geoff Leach, Fabio Zambetta, Pacific Graphics 2013, 41-46
- An Adaptive Octree Grid for GPU-based Collision Detectionof Deformable Objects, Tsz Ho Wong, Geoff Leach, Fabio Zambetta, *submitted* to Computer Graphics International 2014

The thesis was written in the `TexMaker` editor on Windows 7, and typeset using the LaTeX 2_ε document preparation system.
All trademarks are the property of their respective owners.

Contents

List of Figures

List of Tables

Abstract

Physically based cloth simulation in computer graphics has come a long way since the 1980s. Although extensive methods have been developed, physically based cloth animation remains challenging in a number of aspects, including the efficient simulation of complex internal dynamics, better performance and the generation of more effects of friction in collisions, to name but a few. These opportunities motivate the work presented in this thesis to improve on current state of the art in cloth simulation by proposing methods for cloth bending deformation simulation, collision detection and friction in collision response.

The structure of the thesis is as follows.

A literature review of work related to physically based cloth simulation including aspects of internal dynamics, collision handling and GPU computing for cloth simulation is given in Chapter 2.

In order to provide a basis for understanding of the work of the subsequent chapters of the thesis, Chapter 3 describes and discusses main components of our physically based cloth simulation framework which can be seen as the basis of our developments, as methods presented in the following chapters use this framework.

Chapter 4 presents an approach that effectively models cloth non-linear features in bending behaviour, such as energy dissipation, plasticity and fatigue weakening. This is achieved by a simple mathematical approximation to an ideal hysteresis loop at a high level, while in textile research bending non-linearity is computed using complex internal friction models at the geometric structure level.

Due to cloth flexibility and the large quantity of triangles, in a robust cloth system collision detection is the most time consuming task. The approach proposed in Chapter 5 improves performance of collision detection using a GPU-based approach employing spatial subdivision. It addresses a common issue, uneven triangle sizes, which can easily impair the spatial subdivision efficiency. To achieve this, a virtual subdivision scheme with a uniform

grid is used to virtually subdivide large triangles, resulting in a more appropriate cell size and thus a more efficient subdivision.

The other common issue that limits the subdivision efficiency is uneven triangle spatial distributions, and is difficult to tackle via uniform grids because areas with different triangle densities may require different cell sizes. In order to address this problem, Chapter 6 shows how to build an octree grid to adaptively partition space according to triangle spatial distribution on a GPU, which delivers further improvements in the performance of collision detection.

Friction is an important component in collision response. Frictional effects include phenomena that are velocity dependent, such as stiction, Stribeck friction, viscous friction and the stick-slip phenomenon, which are not modelled by the classic Coulomb friction model adopted by existing cloth systems. Chapter 7 reports a more comprehensive friction model to capture these additional effects.

Chapter 8 concludes this thesis and briefly discusses potential avenues for future work.

Chapter 1

Introduction

Reproducing realistic animation has been a major research goal in computer graphics from the outset. For both off-line and real-time systems, physical accuracy is a vital criterion that determines the quality of realism. Initially, simple physics laws, e.g. gravity, were applied for very basic needs. For the motion of complex bodies, artists had to manually specify objects' behaviour frame by frame, giving limited results. However hardware capabilities have improved at an exponential rate, which has dramatically enhanced the performance of computers doing physics computation and has led to the development of more and more complicated physical models. Thanks to this rapid growth of computational power, physically based simulation has become a standard approach to improve realism, and has been gaining an increasing amount of attention from computer graphics researchers for aspects including fluid dynamics, rigid, deformable and cloth objects simulation. Many techniques have been developed to generate fascinating physical effects that significantly enhance the visual quality and the richness of virtual scenarios. Unsurprisingly, the trend of development in physically based animation will continue due to the increasing use of physics in a wide range of applications, such as virtual training systems, video games as well as special effects in movies, and so on.

Apart from computer graphics, physics simulation has expanded tremendously in fields of computational sciences, including mechanical engineering, material engineering and mathematics, to name but a few. In fact, research on physics in these fields started much earlier, and usually the techniques were then adopted by computer graphics. However their purposes are quite different, so they each focus on different aspects. For example, construction simulation for parametric analysis of forces and stress, vehicle crash simulation for composite

materials, or friction simulation for mechanical tribology, are concerned with the precision and accuracy of physics simulation. Clearly, such fidelity comes at a high computational cost. On the other hand, physically based simulation in computer graphics generally aims to produce visually realistic or acceptable results. So, apart from visual realism, computational efficiency, performance, stability and controllability are also important, especially for real-time and interactive applications. In other words, it is crucial to strike a reasonable balance between physical accuracy and computational efficiency.

One of the most important materials to model in a virtual scene is cloth. Cloth objects are ubiquitous in our daily life, since we are constantly in contact with garments, bedclothes, table sheets, towels, curtains and many other fabric products. The textile research community was established in 1930s. Because their primary research purpose is to support manufacturing in the textile industry, the textile community has been particularly concerned with the weave structure and the measurement of properties of fabric materials, which enables the development of methods for modelling and predicting complex fabric behaviour. A half-century later, physically based cloth simulation was first introduced into computer graphics with the goal of producing images and animations that look as physically realistic as possible within the limits of computational efficiency. Since then cloth simulation has become increasingly important for video games, movies and even virtual fashion applications. This thesis focuses on cloth simulation including its internal dynamics and collision handling, and reports improvements to both physical effects and performance aspects.

Recently, graphics processing units (GPUs) have become a powerful tool to accelerate intensive computation for a wide range of applications, including physics simulation. Originally, GPUs were designed to compute graphics. Later, due to the highly parallelism, graphics shading languages were used to handle general purpose computing. Recently, specialised platforms and programming models are developed to more easily use GPUs as generalised computing devices. A part of the work presented in this thesis is the use of GPU computing to gain better performance.

1.1 Motivation

This thesis is motivated by several factors: effectively modelling non-linear bending behaviour using hysteresis, improving performance of collision detection and the modelling of additional effects of friction in collision response, which are discussed in the order in which they were developed as components of the cloth simulation framework.

- **Modelling Non-linear Bending Behaviour**: Due to its thin structure, cloth exhibits a very low bending rigidity. Therefore, a compressive force applied at cloth easily leads to out-of-plane movements forming dynamic, complex wrinkles and folds, which makes modelling bending deformation a key aspect for realistic cloth simulation. However, efficient modelling of realistic bending behaviour remains a challenge. Cloth bending behaviour shows high non-linearity which in textile research is described by a rheological model that contains a generalized bending rigidity and bending hysteresis [Chapman, 1975b; Bliman and Sorine, 1995a]. The latter physically describes nonlinear deformation and effects. Although many existing cloth modelling systems can generate good looking wrinkles and folds, hysteresis has received relatively little attention in computer cloth simulation, and we report the first use of hysteresis to physically capture non-linear bending effects that we are aware of. From a practical point of view, fully accurate modelling these effects is computationally expensive and unnecessary. Therefore an approximation method is needed. We propose an efficient method to simulate such features with limited computational costs using a simple mathematical approximation of hysteresis.

- **Performance in Collision Detection**: Collision detection is an intensive and challenging task in physics simulation. It is also the main performance bottle-neck in most cloth systems, due to cloth flexibility that gives rise to a huge number of self-intersections and collisions between cloth and objects. Spatial subdivision is a common technique to address this issue. Good subdivision reduces an $O(n^2)$ problem to an average complexity of $O(n)$ in a scene containing n triangles. However, duplicate tests are easily generated when triangle pairs are assigned to multiple cells. Grand [Grand, 2007] addresses this problem on a GPU using control bits. Unfortunately, this method requires a large cell size, which severely limits subdivision efficiency, especially when triangle sizes are uneven. In fact, it is unrealistic to expect a mesh with evenly sized triangles. To efficiently address this problem and still use control bits to remove duplicate tests, we present a GPU-grid based approach that *virtually* subdivides large triangles with a uniform grid to enable the use of a smaller, better suited cell size.

  Apart from the problem of uneven triangle sizes, spatial subdivision methods suffer from another common issue: uneven triangle spatial distributions, which can also lead to an inefficient subdivision. It is difficult for it to be solved by uniform grids, because a uniform cell size can be too small for low density areas, and too big for high density areas. In contrast, for these uneven cases, non-uniform subdivision works better. Techniques that

non-uniformly partition space mainly include hierarchical grids and octrees. Hierarchical grids have been adopted by previous GPU-based algorithms, due to their simplicity and continuity in memory. However, octrees offer a better adaptation to spatial distributions. We present a method using an octree grid that takes a middle path between these two structures to accelerate collision detection.

- **Modelling Additional Effects of Friction**: Once collisions have been detected, they have to be resolved to generate physically correct motion, which is referred to as collision response. Friction is an important component during this process. Such complex phenomenon that resists tangential motion between two contacting objects can produce stable folds, improving the visual richness. In computer graphics existing physics simulations typically use a simple Coulomb model However it is limited since it intends to only capture fundamental frictional phenomena. Many observed friction effects such as stiction, Stribeck friction, viscous friction and the stick-slip phenomenon are not modelled by it. These frictional phenomena are observed under both dry and wet conditions in fabrics that deform viscoelastically. Therefore a more comprehensive friction model to consider these additional effects is worth investigating. To achieve this, we describe an improved physically based friction model that extends the Coulomb model.

1.2 Contributions and Overview

In this section, we provide an overview of this thesis and highlight its contributions.

- **Related Work** Previous work related to cloth internal dynamics, collision detection, collision response and GPU computing for cloth simulation is discussed in Chapter 2.
- **Cloth Simulation Framework**: Before presenting the main contributions, the framework which developments presented in this thesis use, is presented in Chapter 3 to provide a basis for understanding of how cloth simulation works. Internal dynamics are described first, including three mechanical properties: stretching, shearing, and bending. Although there are a number of different approaches for modelling these deformations, they all require numerical integration of differential equations governing the dynamics. Therefore several numerical integration methods solving the differential equations system are discussed. Finally techniques for collision detection and collision response are summarised.
- **Modelling Bending Behaviour Using Hysteresis**: In Chapter 4, a novel simple and fast method is presented to simulate complex nonlinear bending behaviours of cloth objects,

and bending effects including plasticity and energy dissipation [Wong et al., 2013b]. The primary idea is to use a mathematical approximation to an ideal hysteresis loop at a high level without complex internal friction models at the fabric geometric structure level. The approximation describes two different curves governing the curvature moment relationship for bending and recovery to look up bending and unbending forces, which include non-linear bending stiffness. Previous methods to model plasticity typically use approaches where bending is treated linearly and an elastic threshold is used to trigger plastic deformation, hence non-linear bending behaviour and effects caused by energy dissipation cannot be fully captured. In contrast, our method inherently captures such effects, because hysteresis features are included in the approximation. Further, by extending this model, fatigue weakening that turns increasingly residual wrinkles into permanent is easily simulated.

- **Virtual Subdivision for GPU based Collision Detection Using a Uniform Grid**: In Chapter 5, we present a simple uniform grid based collision detection method on GPUs that uses a better cell size by virtually subdividing large triangles into small child triangles [Wong et al., 2012]. Not only does this address the issue of uneven triangle sizes to dramatically reduce the number of tests, but it also takes advantage of control bits to quickly remove duplicate tests. Child triangles are virtual because they are only used for the purpose of broad phase collision detection tests, and the newly generated vertices and edges are not used in cloth internal dynamics, narrow phase tests or collision response. Most duplicate tests caused by the virtual subdivision scheme are avoided in the broad phase stage by a lightweight removal scheme. Experiments comparing results with those of prior methods show that performance improvements are achieved.

- **An Adaptive Octree Grid for GPU based Collision Detection**: Although uneven triangle sizes can be addressed by a virtual subdivision scheme with a uniform grid, uneven spatial distributions can also significantly impact the subdivision efficiency and this problem is difficult for uniform grids. To tackle this problem, we explore a GPU-based approach using an adaptive octree grid [Wong et al., 2014] in Chapter 6. The octree grid combines advantages of hierarchical grids and octrees: the grid can be more efficiently processed on GPUs, and the octree partitions space adapting better to uneven triangle spatial distributions, obtaining a considerable reduction in the number of broad phase tests. Since performing a large number of tests in the broad phase dominates the total running time, it is worth further reducing calculations in this step. To achieve this, a two-stage scheme is used to minimise the number of tests by optimising the octree subdivision. This method is

also able to address the issue of uneven triangle sizes, as this can often be seen as uneven spatial distributions.

- **An Improved Friction Model**: Chapter 7 describes an improved physically based friction model for collision response in cloth simulation [Wong et al., 2013a]. In order to capture addition velocity-dependent effects that are not modelled by existing cloth systems, the model borrows theories from tribology research to extend the Coulomb model. The integration of the extended friction model in two commonly adopted contact methods: the constraint handling approach and the velocity filter framework, is discussed.
- **Conclusion**: Conclusion and potential future work follow in Chapter 8.

Chapter 2

Related Work

This chapter provides an overview of previous work related to cloth internal dynamics, collision handling and GPU computing for cloth simulation.

2.1 Cloth Dynamics

Cloth simulation in computer graphics started in the 1980s. The earliest work is in [Weil, 1986] where folding effects of a free-hanging cloth supported by a number of constraint points are approximated for static images, which is a geometric method. This kind of approaches approximate cloth behaviour using computational geometry techniques without mechanical information. However, in this thesis we will only focus on physically based approaches. Generally most of physical approaches to date treat cloth as a 2D continuum with no thickness. Although some methods allow modeling of cloth thickness such as [Bridson et al., 2002; Selle et al., 2009], it is not considered in the internal dynamics computation but just for the treatment of collision response. An overview of cloth internal dynamics are given in this section along with time integration work. It is broken down into in-plane and out-of-plane deformation (Figure 2.1).

2.1.1 In-plane Deformation

We broadly follow [Volino et al., 2009] to categorize in-plane force approaches further into particle systems and finite element methods.

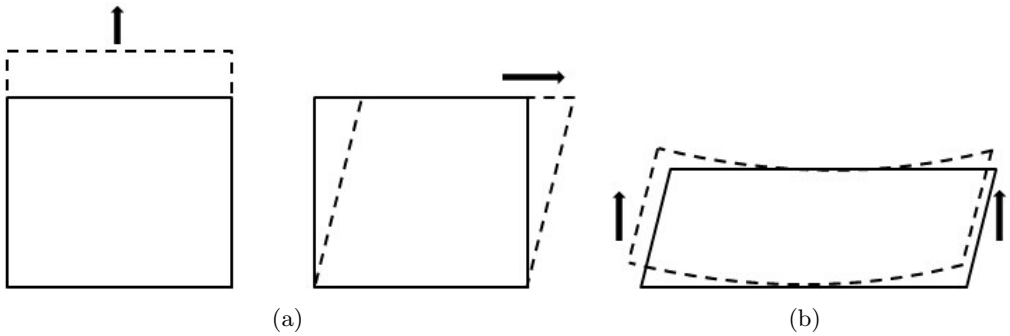


Figure 2.1: (a) In-plane deformation (b) Out-of-plane deformation

Particle Systems

Breen et al. [1992] deal with cloth as interacting particles based on grid to model the mechanical constraints between the threads in a woven plain. In order to model specific kinds of woven cloth, they extend the model to use empirical data measured by Kawabata system, and thus be able to consider nonlinear behaviour including hysteresis effects [Breen et al., 1994]. These methods are only designed to compute static draping results. Later, by extending the particle system, Eberhardt et al. [1996] introduce an approach to simulate dynamic behavior over time. However these grid-based particle approaches are computationally expensive.

Probably the simplest way to physically simulate cloth objects is the spring-mass based particle system introduced by Provot [1996] where simple linear springs are used in a regular rectangular mesh to represent stretching, shearing and bending forces. A strain rate limiting scheme is proposed to remove bad strain rates, resulting in better results in cloth behaviour and explicit time integration performance. This strain limiting scheme is extended in a velocity filter framework [Bridson et al., 2002]. Baraff and Witkin [1998] describe a method working with arbitrary triangle meshes. In-plane force functions and derivatives are derived from internal energy functions formed by two different continuum formulation for stretching and shearing respectively. A big advance of their work is that an implicit integration method is used to address numerical computation issues found in explicit integration methods, leading to a stable simulation with large time steps. By following [Provot, 1996], Choi and Ko [2002] treat in-plane deformation as simple linear spring model, while the force is derived from an energy function. A second-order backward difference integrator is used for purposes of accuracy and to reduce artificial damping. Other works exploiting advanced different integration methods include [Bridson et al., 2003a; Volino and Magnenat-Thalmann, 2005a;

Schroeder et al., 2011].

The main issue that particle approaches suffer is inaccuracy in in-plane elasticity, especially for large deformations. Although methods exist that carefully adjust values of spring stiffness to address this problem to an extent [Van Gelder, 1998; Lloyd et al., 2007], the high dependencies on mesh topology and modelling systems make them unwieldy to use. Volino and Magnenat-Thalmann [2005b] describe a method that obtains relatively accurate strain-stress relationship by polynomially approximating strain-stress curves, without losing the simplicity of spring-mass systems. Later Volino et al. [2009] improve on this method to model nonlinear in-plane deformations by using the nonlinear Green-Lagrange tensor which makes the method equivalent to finite element methods, offering a highly accurate and easy-to-implement cloth modelling system.

Finite Element Methods

An alternative way to accurately model cloth is finite element methods. In general, continuous objects modelled by continuum mechanics are discretised into a finite set of elements to allow solutions that can be solved analytically. These methods are very common and popular to solid objects modelling in mechanical engineering. Compared to most particle approaches, an important advantage is that the body volume (area for cloth) is considered in formula, therefore behavior is independent of a specific mesh. Unsurprisingly finite element methods require more computation time.

Terzopoulos et al. [1987] introduce this technique to computer graphics for the purpose of modelling elastic deformable objects including cloth. Modelling of inelasticity is added in their later work [Terzopoulos and Fleischer, 1988] where objects experience inelastic deformation when the applied force exceeds a limit. Although a number of works have been proposed to reduce computation time required by Finite Element Methods [Debunne et al., 2001; Wu et al., 2001], accurate real-time simulation has yet to be achieved, especially for large rotational deformations, which require higher accuracy, due to their non-linear nature. Müller et al. [2002]; Müller and Gross [2004] handle the rotational component separately by warping the stiffness matrix based on a tensor field that describes local rotations using a local coordinate frame. Etzmubeta et al. [2003] apply this technique to cloth simulation.

Other Methods

There are a number of works that do not belong to the two categories discussed above. Müller et al. [2007] propose a position based dynamic system that works directly on positions and omits the velocity layer. This method is sped up in [Müller, 2008] where a multigrid scheme is utilized. There are also few methods designed to model knitted fabric at the yarn level [Kaldor et al., 2008; 2010; Yuksel et al., 2012].

2.1.2 Out-of-Plane Deformation

The bending force generated in bending deformation to resist flexion is usually treated as a out-of-plane force.

In the simple linear spring-mass system [Provot, 1996], bending force is formed by a spring connected to opposite corners of adjacent triangles. Its force computation is akin to stretching and shearing springs. Choi and Ko [2002] extend this work to improve the realism of wrinkles and overcome the so called post-buckling instability without introducing damping forces. Since these techniques do not consider the angle of adjacent triangles, accurate deformation sensitive to curvatures is hard to simulate.

Baraff and Witkin [1998] derives the bending force from an energy function that takes account of the dihedral angle between two adjacent triangles. A simple and fast bending model [Bridson et al., 2003a] also measures the dihedral angle. Grinspun et al. [2003] use mean curvatures instead of Gaussian curvatures for membrane simulation. Since not only the dihedral angle but also triangle heights and the length of the common edge of adjacent triangles are considered into their formula, reasonably resolution-independent bending behaviours are obtained.

In general, bending behaviour is highly nonlinear. By extending the work [Bergou et al., 2006] where bending energy is quadratic in positions, Garg et al. [2007] expose the nonlinear nature of bending energies by establishing a cubic hinge (a pair of edge-adjacent triangles). Complex derivation caused by the nonlinearity is avoided via an approximate Newtons solver with precomputation of Jacobian matrix. In order to capture effects of seams, Pabst et al. [2008] use measured data describing the curvaturemoment relationship from textile research to model nonlinear bending behaviour. The bending model described in Chapter 4 is an improvement on their approach. Nonlinear bending forces have also been discussed in [Terzopoulos et al., 1987; Simo and Fox, 1989; Grinspun et al., 2002]. Other approaches using measured data to model nonlinear behaviours include [Breen et al., 1994; Eberhardt et al.,

1996; Eischen and Bigliani, 2000].

Bending Hysteresis

Bending hysteresis physically describes the high bending non-linearity and bending energy dissipation due to internal friction which also gives rise to plasticity. Terzopoulos and Fleischer [1988] first introduced plasticity to the computer graphics community. The earliest work to consider hysteresis is [Breen et al., 1994]. In a report by [Ngoc and Boivin, 2004], relatively complex friction terms are used upon a spring-mass system with a regular rectangular mesh to accurately capture non-linear traction, shearing and bending behaviours including hysteresis for the purpose of parameters identification. Plasticity effects are suggested as a future work in this report. Kim et al. [2011] take plastic effects as energy loss over time, while firm physical foundation and explanation are missing.

2.2 Collision Handling

Generally, collision handling is the main bottle-neck in cloth simulation in terms of both performance and robustness. It involves many research challenges, and has been studied intensively for decades. This problem normally is broken down into two sub-problems, collision detection and collision response.

2.2.1 Collision Detection

Collision detection in computer graphics has a rich history. Earlier research mainly focuses on rigid objects, while the last two decades have seen increasing attention to deformable objects including cloth. We can not possibly cover all previous work in this thesis. We refer interested readers to [Teschner et al., 2004] for more detailed information. Most efficient algorithms reduce the computation using a broad phase where primitive pairs that apparently do not collide with each other are quickly eliminated. Once the broad phase is done, a narrow phase is responsible for more expensive elementary tests computing exact intersection information of vertex-triangle pairs and edge-edge pairs for collision response.

Bounding volume hierarchies (BVH) have long been used to accelerate collision detection, and have proven to be a very efficient technique in the broad phase. Its key idea is to recursively partition the mesh to construct a tree where each node contains a subset of primitives enclosed by a bounding volume (BV). Usually the leaf node contains a single primitive. By traversing the tree in a top-down order, pairs of leaf nodes with BVs overlapped

are found and registered for further elementary tests and intersection tests if necessary. During this process many unnecessary tests can be quickly pruned.

Large efforts have gone into exploiting different types of BV including spheres [Palmer and Grimsdale, 1995], axis aligned bounding boxes (AABBs) [van den Bergen et al., 1998; Bridson et al., 2002; Selle et al., 2009], oriented bounding boxs (OBBs) [Gottschalk et al., 1996], k-discrete oriented polytopes (DOPs) [Klosowski et al., 1998; Fnfzig and Fellner, 2003] and convex hulls [Ehmann and Lin, 2001]. In general each BV type has its own advantages and disadvantages, and a trade-off exists between culling efficiency and BV test costs. The better choice varies depending on specific environments and the purpose of any given application.

In some works, the construction of BVH is a preprocessing step before simulation starts. The tree structure is constructed once for meshes under undeformed state, and conserved during the whole simulation except for some special cases such as tearing, which changes mesh topology. At a broad level there are two common ways to construct the BVH structure, top-down schemes [Palmer and Grimsdale, 1995; Gottschalk et al., 1996] and bottom-top schemes [Bridson et al., 2002; Selle et al., 2009], both of which are essentially recursive. Some other works choose to build the hierarchy for the whole scene, so as to update it in every time step. Instead of simply rebuilding hierarchies, many efficient updating techniques have been proposed to accelerate this step [Thomas Larsson, 2001; Mezger et al., 2003; Larsson and Akenine-Möller, 2006; Otaduy et al., 2007].

An alternative way of efficiently removing unnecessary tests in the broad phase is referred to as a spatial subdivision scheme. Its main strategy is to partition space into many cells, and then perform collision detection tests only for triangle pairs belonging to the same cell, resulting in an average complexity of $O(n)$ in a scene containing n triangles.

The most straight-forward way to partition space is a uniform grid [Zhang and Yuen, 2000; Teschner et al., 2003]. The performance strongly relies on the cell size, because a large cell size may result in too many triangles belonging to the same cell, or triangles may overlap too many cells if the cell size is small. Therefore, efforts have been directed at determining an appropriate cell size. Teschner et al. [2003] choose a cell size 1.2 times the average primitive radius. Unfortunately, an appropriate cell size is difficult to find or may even not exist if triangle sizes are uneven. To address this problem, a hierarchical grid with multiple cell levels is adopted [Schornbaum, 2009; Eitz and Lixu, 2007], where primitives are assigned to different levels according to its size.

Another issue is that duplicate tests can be easily introduced due to triangle pairs assigned to multiple cells. Grand [2007] effectively solves this problem using the control bits scheme for

collision detection of spheres on GPUs. Pabst et al. [2010] extend this method for deformable triangular meshes. However, the control bits scheme requires the cell size greater than the bounding volume of the largest triangle, which severely limits the subdivision efficiency. In order to address the uneven triangle size problem and still use control bits to remove duplicate tests, a GPU-grid based approach that virtually subdivides large triangles with a uniform grid is presented in Chapter 5 to enable the use of a better suited cell size.

Spatial subdivision also suffers from the issue of uneven triangle spatial distributions, which is difficult for uniform girds, because a uniform cell size can be too small for low density areas, and too large for high density areas, giving an inefficient subdivision. Alcantara et al. [2009] and Fan et al. [2011] partially solve this issue by subdividing dense cells into eight child cells using a two-level hierarchical grid. Methods adopting a hierarchical grid with multiple cell levels [Schornbaum, 2009; Eitz and Lixu, 2007] can also alleviate this problem to an extension, but lacks the adaptability to triangle spatial distributions to reduce the number of tests as much as possible. A GPU-based method using an octree grid is presented in Chapter 6 to better address this problem.

Many techniques that are not limited to a specific framework (e.g. BVHs based or grid based) are proposed to improve the performance. A large number of duplicated elementary tests can be generated simply because a vertex and edge might belong to multiple triangles. Curtis et al. [2008] address this issue by assigning each vertex and edge to only one triangle, therefore tests of triangle pairs not possessing involved vertices or edges are ignored. Other methods solving the same problem include a randomized marking scheme [Wong and Baciu, 2006] and orphan sets [Tang et al., 2008]. Tang et al. [2008] point out another issue: due to the adjacency of triangles, a considerable number of self collision BV tests are processed even if there is no collision at all. They avoid these unnecessary tests by introducing a step, before the broad phase, that measures continuous normal cones to determine whether the remaining steps can be skipped. To further increase efficiency, a filter is inserted between the broad phase and narrow phase to remove false positives that cannot be culled by BV tests [Tang et al., 2010a].

The task of elementary tests in the narrow phase is to output exact intersection information such as contact time, points and normals. Moore and Wilhelms [1988] propose a fifth-order polynomial to compute such information for vertex-triangle intersections. Provot [1997] reduces the computation to a three order polynomial. Edge-edge intersections are also computed, giving better results. The industry adopted framework [Bridson et al., 2002] improves on this work to interpolate impulses using barycentric coordinates.

2.2.2 Collision Response and Friction

Once collision detection is done physical collision responses have to be generated. Collision response is of the utmost importance for robustness of physics simulation. It can be broadly subdivided into constraint-based approaches, impulse-based methods [Mirtich and Canny, 1995; Mirtich, 1996] and penalty-based methods [Terzopoulos et al., 1987; Wriggers et al., 1990; Tang et al., 2012]. In this thesis we focus on constraint-based approaches since which offer better accuracy in terms of physics.

Provot [1997] first treats collision responses geometrically for cloth simulation. For contacts between cloth and external objects, Baraff and Witkin [1998] enforce a complete constraint that eliminates relative motion and a partial constraint that only allows sliding on the plane proportional to the contact normal force, to achieve stiction and kinetic friction effects, respectively. For self collisions, the framework proposed by [Bridson et al., 2002] guarantees a non-interfering state at the end of every time step. A cloth thickness is included to let cloth feels repulsion forces in successive time steps in the case of self contacts, which allows friction being modelled in subsequent time steps generating more stable and realistic folds. Bridson et al. [2003a] implement a post-processing technique to preserve wrinkles.

Friction is paramount to realism, since it is a ubiquitous phenomenon which occurs whenever two objects are in contact in nature. Since Leonardo da Vinci discovered rules of dry friction, such phenomenon has been studied in many research fields. Classic empirical laws by Amonton and Coulomb are dubbed Coulomb friction. Even today this classic model has been widely used to estimate friction behaviour in many areas including computer graphics. Terzopoulos et al. [1987] is among the earliest works including friction for physics simulation in computer graphcis. Baraff and Witkin [1998] treat cloth-object friction as constraints. Bridson et al. [Bridson et al., 2002] computes friction as impulses. As an extension of this work, Selle et al. [2009] propose a novel friction scheme to obtain more accurate collision response. Pabst et al. [2009] introduce an anisotropic friction model from textile research for contacts between deformable objects including cloth and materials with anisotropic surface. Chen et al. [2013] improve cloth-body contact effects using aerodynamics and friction data captured from real-world. However some important velocity-dependent effects such as stiction, Stribeck friction, viscous friction and the stick-slip phenomenon are not considered by these work. An elaborate friction model that extends the Coulomb model is presented in Chapter 7.

The issue of multiple collisions that resolving collisions can generate new collisions must

be addressed for any robust cloth system. Provot [1997] describe an efficient method called "impact zones" to overcome this problem. After a fix number of iteration, if converge is not guaranteed vertices involved in multiple collisions are extracted as "impact zones" which are simulated as rigid objects and grow until no more collisions are detected. Bridson et al. [2002] adopt this strategy and correct the formula for angular velocity. However this technique is just physics plausible. Harmon et al. [2008] present a projection method to resolve multiple collisions while preserve physical accuracy as much as possible.

2.3 Parallel Computation for Cloth Simulation on GPU

As graphics processing units evolved, GPU computing has been showing its potential to simulate physics over the last few years.

Early days shaders were utilised to do general purpose computing on GPUs, which is referred to as GPGPU. Green [2004] presents the first GPU-based cloth simulation. Later, Zeller [2005] improves on Greens work by taking advantage of geometry shaders. Rodrïguez-Navarro and Susïn [2006] implement a finite element method algorithm evolved by an implicit integrator on GPU. Other shader based cloth simulation methods include [Rodrïguez-Navarro et al., 2005; Tejada and Ertl, 2005; Vassilev and Spanlang, 2012]. Shaders have been also used to accelerate matrix operations [Larsen and McAllister, 2001; Thompson et al., 2002], resulting in better performance for numerical simulation [Rumpf and Strzodka, 2001a;b; Bolz et al., 2003; Krüger and Westermann, 2003]. Shader based collision handling methods use depth images to detect contacts therefore are not physically based, hence not relevant to this thesis.

However, shaders are designed for graphics applications. In order to more easily take advantages of GPUs' parallelism, specialised GPU computing languages were designed, such as CUDA [NVIDIA] and OpenCL [Khronos].Rasmusson et al. [2008] present a CUDA-based implementation of spring-mass systems. Comas et al. [2008] develop a CUDA-based nonlinear finite element model to model soft tissue for medical purposes. Li et al. [2011] intend to achieve real-time cloth animation by implementing a CPU-GPU hybrid computing framework. A GPU-based framework able to accurately simulate fabric behaviour including nonlinear strain-stress relationships is presented in [Kevelham and Magnenat-Thalmann, 2012]. Tang et al. [2013] simulate high-resolution cloth objects using a GPU-based streaming algorithm. Methods taking advantages of the CUDA architecture to accelerate matrix operations are discussed in [Bell and Garland, 2008; 2009; Wang et al., 2010; Feng et al., 2011].

For collision handling, collision response has not yet been implemented on the GPU completely due to its successiveness, while the focus on collision detection in recent years has shifted to GPU computing algorithms using CUDA, and gained considerable increases in performance. Similar to the traditional CPU-based counterparts, GPU-based collision detection approaches are categorized into BVHs methods [Kim et al., 2009; Lauterbach et al., 2009; 2010; Tang et al., 2011] and spatial subdivision methods [Grand, 2007; Pabst et al., 2010; Fan et al., 2011].

Chapter 3

Cloth Simulation Framework

Components of our physically based cloth simulation framework are described in this chapter to provide an overview and a basis for understanding of the remaining chapters of the thesis. Developments presented in the following chapters are built on this framework.

Cloth behaviour and appearance are crucially determined by the modelling of internal forces. In-plane forces resist stretch and shear deformation, while formation of folds and wrinkles mainly depends on out-of-plane forces. As an indispensable part of physical simulation, time integration not only governs how simulation evolve but also significantly affects the whole dynamic behaviour.

Fabric is usually designed to cover human body or furniture such as tables and beds. It is necessary to address collisions between cloth and external objects. Nevertheless, the main challenge in collision handling for thin, flexible cloth objects is self collision. A robust cloth system must detect and solve all collisions, otherwise missed collisions may give rise to penetrations. This is a severe problem that can not be tolerated for cloth simulation, because such penetrations are highly visible due to the thinness of cloth, and can lead to permanent tangles giving physically incorrect results.

The remainder of this chapter is organised as follows. Two in-plane force models and a bending force model with their computation are presented in Section 3.1. Section 3.2 describes time integration methods including explicit and implicit solvers along with force computation. The framework of collision handling is given in Section 3.3.

3.1 Cloth Internal Force

In this section, computation of in-plane forces and out-of-plane forces is described.

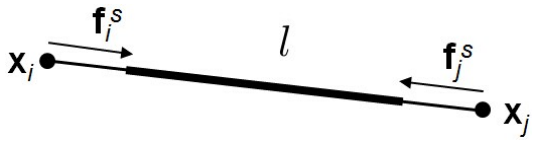


Figure 3.1: Compute internal forces using a linear spring

3.1.1 In-plane Force

The simplest way to compute internal forces is a spring-mass system using simple linear springs [Provot, 1996]. Although this model is quite simple, it appears to satisfy visual requirements for many applications.

The cloth mesh is composed of a set of discrete particles grouped into triangles. The mass, position and velocity of the ith particle are m_i, $\mathbf{x}_i$ and $\mathbf{v}_i$, with $i \in [1, N]$ where N is the number of particles. Particles are connected by a set of springs (triangle edges). Forces along such springs describe stretch forces, while shear forces are ignored in this model. The spring forces acting on particle i and j connected by a spring (Figure 3.1) are

$$\begin{aligned} \mathbf{f}_i^s &= k_s \frac{\mathbf{x}_j - \mathbf{x}_i}{|\mathbf{x}_j - \mathbf{x}_i|}(|\mathbf{x}_j - \mathbf{x}_i| - l) \\ \mathbf{f}_j^s &= \text{-}\mathbf{f}_i^s \end{aligned}, \tag{3.1}$$

where k_s is the spring stiffness, and l the rest length of the spring. The forces depend on positions only. They are proportional to the difference between the current spring length and its initial length in the undeformed state. The damping forces that depend on both position and velocity acting on these two particles are

$$\begin{aligned} \mathbf{f}_i^d &= k_d(\mathbf{v}_j - \mathbf{v}_i)\frac{\mathbf{x}_j - \mathbf{x}_i}{|\mathbf{x}_j - \mathbf{x}_i|} \\ \mathbf{f}_j^d &= \text{-}\mathbf{f}_i^d \end{aligned} \tag{3.2}$$

where k_d is the damping stiffness. The sum of the spring force and damping force

$$\begin{aligned} \mathbf{f}_i &= \mathbf{f}_i^s + \mathbf{f}_i^d \\ \mathbf{f}_j &= \text{-}\mathbf{f}_i \end{aligned}$$

forms the in-plane forces. These forces are momentum conserving.

Instead of computing forces directly from the spring length, forces can be also derived

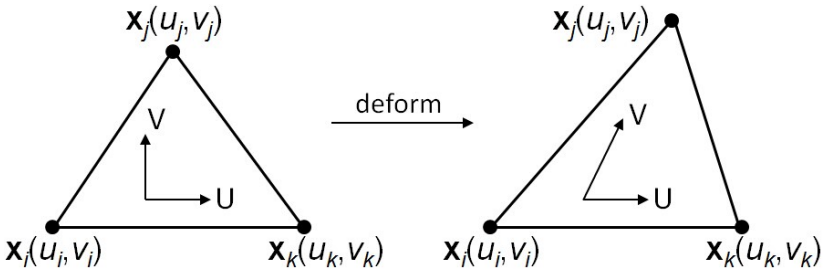


Figure 3.2: A triangle is assigned fixed 2D plane coordinates, and is deformed from its initial state in 3D space.

from a scalar energy function $E(\mathbf{x})$ based on constraints $C(\mathbf{x})$:

$$E(\mathbf{x}) = \frac{1}{2} k C(\mathbf{x})^2, \tag{3.3}$$

$$\mathbf{f} = -\frac{\partial E}{\partial \mathbf{x}} = -kC\frac{\partial C}{\partial \mathbf{x}}, \tag{3.4}$$

where k is the stiffness. The energy is zero when the mesh is undeformed. In other words, the energy is larger than zero if the mesh is deformed. The negative gradient of energy is the direction of the force, which means that internal forces exerted on the mesh act to recover the deformation. Now the problem is to define the constraint $C(\mathbf{x})$.

To construct stretch energy, a simple constraint based on the spring length can be used. However, we follow [Baraff and Witkin, 1998] where continuum mechanics and all three particles of a triangle are considered to compute stretch and shear forces for better results.

At first, each particle is assigned a fixed 2D plane coordinate (u_i, u_j) (Figure 3.2), and assuming a C^1 continuous function $\mathbf{w}(u_i, u_j)$ maps plane coordinates to the current position $\mathbf{x}_i$ in world space, so that stretch or compression in the u and v directions can be measured by the magnitude of derivatives $\mathbf{w}_u = \partial\mathbf{w}/\partial u$ and $\mathbf{w}_v = \partial\mathbf{w}/\partial v$ at any point of the mesh. Similar to what happens in continuum mechanics, when the mesh is underfomed $|\mathbf{w}_u| = |\mathbf{w}_v| = 1$ and $\mathbf{w}_u^T\mathbf{w}_v = \mathbf{w}_v^T\mathbf{w}_u = 0$, indicating that axes u and v have unit length (no stretch or compression) and are perpendicular to each other (no shear). This can be written as

$$\begin{bmatrix} \mathbf{w}_u^T \\ \mathbf{w}_v^T \end{bmatrix} \begin{bmatrix} \mathbf{w}_u & \mathbf{w}_v \end{bmatrix} = \begin{bmatrix} |\mathbf{w}_u|^2 & \mathbf{w}_u^T\mathbf{w}_v \\ \mathbf{w}_v^T\mathbf{w}_u & |\mathbf{w}_v|^2 \end{bmatrix}. \tag{3.5}$$

To address the question of how to apply this general continuous vector field to a discrete

triangle mesh, the following scheme is used. For a given triangle with particles i, j and k, let

$$\begin{aligned} \Delta\mathbf{x}_1 &= \mathbf{x}_j - \mathbf{x}_i, \qquad \Delta\mathbf{x}_2 = \mathbf{x}_k - \mathbf{x}_i, \\ \Delta u_1 &= u_j - u_i, \qquad \Delta u_2 = u_k - u_i, \\ \Delta v_1 &= v_j - v_i, \qquad \Delta v_2 = v_k - v_i \end{aligned}$$

and assume that $\mathbf{w}(u_i,\ u_j)$ is a linear function, thus derivatives $\mathbf{w}_u$ and $\mathbf{w}_v$ are constant. Hence within a triangle

$$\Delta\mathbf{x}_1 = \mathbf{w}_u\Delta u_1 + \mathbf{w}_v\Delta v_1, \quad \Delta\mathbf{x}_2 = \mathbf{w}_u\Delta u_2 + \mathbf{w}_v\Delta v_2.$$

Solving this equation, we have:

$$[\mathbf{w}_u \quad \mathbf{w}_v] = [\Delta\mathbf{x}_1 \quad \Delta\mathbf{x}_2]\begin{bmatrix}\Delta u_1 & \Delta u_2 \\ \Delta v_1 & \Delta v_2\end{bmatrix}^{-1}.$$

Therefore the stretch constraint C is a vector and can be seen as a function of particle positions of a triangle with an area a in the undeformed state:

$$\mathbf{C}(\mathbf{x}) = a\begin{bmatrix}|\mathbf{w}_u| - 1 \\ |\mathbf{w}_v| - 1\end{bmatrix}. \tag{3.6}$$

Again, according to continuum mechanics, the shearing deformation can be measured by off-diagonal entries in Equation 3.5, the inner products $\mathbf{w}_u^T\mathbf{w}_v$ and $\mathbf{w}_v^T\mathbf{w}_u$, which must be zero when the mesh is not sheared, because axes u and v are perpendicular to each other. Therefore the constraint for shear forces is:

$$\mathbf{C}(\mathbf{x}) = a\mathbf{w}_u^T\mathbf{w}_v. \tag{3.7}$$

Angles between triangle edges are taken into account by the inner product. By plugging constraints 3.6 and 3.7 into Equation 3.4, stretch and shear forces can be obtained.

Stretch and shear energy are functions of position only (as well as bending energy, presented in the next section), whereas the damping force is a function of both position and velocity. The damping force is defined in terms of constraints described above, and should

act on the opposite direction of $\partial \mathbf{C}(\mathbf{x})/\partial \mathbf{x}$ to damp the velocity. It can be defined as:

$$\mathbf{f} = -\frac{\partial E}{\partial \mathbf{v}} = -k_d \frac{\partial \mathbf{C}(\mathbf{x})}{\partial \mathbf{x}} \frac{\partial \mathbf{C}(\mathbf{x})}{\partial \mathbf{x}}^T \dot{\mathbf{x}}. \tag{3.8}$$

Similar to stretch and shear forces, the damping force arising from each type of energy is derived by plugging each type of constraint into this equation.

3.1.2 Out-of-Plane Force

Both methods discussed above can be extended to model bending forces. In the first method, the linear spring that connects two particles belonging to the same triangle is responsible for the stretch force. Similarly, the bending force can be modelled by a spring that connects unshared particles of adjacent triangles. Since there is no difference in computation between the stretch spring and the bending spring, the computation defined by Equation 3.1 is applied. The second method computes the bending force based on the angle between adjacent triangles. The constraint for bending is:

$$\mathbf{C}(\mathbf{x}) = \theta. \tag{3.9}$$

The angle θ is defined as

$$\theta = sin^{-1}((\mathbf{n}_1 \times \mathbf{n}_2) \cdot \mathbf{e}) \quad \text{and} \quad \theta = cos^{-1}(\mathbf{n}_1 \cdot \mathbf{n}_2),$$

where $\mathbf{n}_1$ and $\mathbf{n}_2$ are unit normals of two adjacent triangles, and $\mathbf{e}$ denotes the unit vector of the common edge. By writing this constraint, the bending force and damping force arising from the bending energy are easily derived using Equation 3.4 and 3.8, respectively.

In order to efficiently capture the nonlinearity exhibited by bending deformations, Chapter 4 presents a method to model bending behaviour using hysteresis.

3.2 Time Integration

To dynamically evolve the system, time integration methods are needed to compute the simulation state at the next time step. They have a significant impact on the accuracy and stability of a simulation. Although large efforts have been made, currently there is no single uniform solution, and this field still continues to attract attentions from researchers.

The notation $\dot{\mathbf{x}}$ and $\ddot{\mathbf{x}}$ is used for the first and second derivatives of position with respect

to time, that is, the velocity $\mathbf{v}$ and acceleration $\mathbf{a}$. All position vectors $\mathbf{x}_i$, $i \in (1, N)$, are combined into a multidimensional position vector $\mathbf{x}$. The same definition is applied to the velocity, acceleration and force vectors. Additionally a diagonal mass matrix $\mathbf{M} \in \mathbb{R}^{3N \times 3N}$ is constructed with diagonal entries $m_1, m_1, m_1, m_2, m_2, m_2, ..., m_N, m_N, m_N$.

3.2.1 Explicit Methods

Explicit integration methods compute the state at the next time step using velocity and acceleration at the current time step. According to Newtons second law of motion,

$$\mathbf{v}(t) = \mathbf{v}(t_0) + \int_{t_0}^{t} \mathbf{M}^{-1}\mathbf{f}(t)dt, \tag{3.10}$$

$$\mathbf{x}(t) = \mathbf{x}(t_0) + \int_{t_0}^{t} \mathbf{v}(t)dt. \tag{3.11}$$

To use numerical integration, these derivatives can be discretely approximated with discrete time step Δt as

$$\mathbf{v}^{t+1} = \mathbf{v}^t + \Delta t \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}^t, \mathbf{v}^t), \tag{3.12}$$

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \Delta t \mathbf{v}^t, \tag{3.13}$$

where superscripts t and $t+1$ indicate the state of the current and next time step respectively, and the force is a function of positions and velocities as described in Section 3.1.1. This simple scheme is the explicit Euler integration method, one of the simplest integrators. However it is not accurate, which can be examined by applying a Taylor series expansion to $\mathbf{x}^{t+1}$:

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \Delta t \dot{\mathbf{x}}^t + \frac{\Delta t^2}{2!}\ddot{\mathbf{x}}^t + \frac{\Delta t^3}{3!}\dddot{\mathbf{x}}^t + \ldots + \frac{\Delta t^n}{n!}\frac{\partial^n \mathbf{x}}{\partial t^n} + \ldots. \tag{3.14}$$

Equation 3.13 is in fact a truncation of the Taylor series, preserving the first two terms at the right side of the expansion. This is to say that the explicit Euler integration achieves a second order accuracy or that the total accumulated error is order $O(\Delta t^2)$, because $\frac{\Delta t^2}{2!}\ddot{\mathbf{x}}^t$ contributes the most to the error term:

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \Delta t \mathbf{v}^t + O(\Delta t^2). \tag{3.15}$$

Such error can easily lead to the issue of instability when the time step size is not small enough or the system is too stiff. Because the force and velocity at the current state are

used to update the system during the entire time step, a large step size increases the error resulting in divergence. For example, a spring defined by Equation 3.1 is stretched to have a length l, the generated spring force intends to recover the deformation to its initial length. If the step size is too large or the spring is too stiff, two particles connected by the spring can cross each other and continue to move to have a new length greater than l, therefore the spring diverges (accumulated errors increase and blows up eventually).

More accurate integration methods can alleviate this problem to a certain extent. The second order Runge-Kutta method is one way to achieve this:

$$\mathbf{v}^{t+\frac{1}{2}} = \mathbf{v}^t + \frac{\Delta t}{2}\mathbf{M}^{-1}\mathbf{f}(\mathbf{x}^t, \mathbf{v}^t), \tag{3.16}$$

$$\mathbf{x}^{t+\frac{1}{2}} = \mathbf{x}^t + \frac{\Delta t}{2}\mathbf{v}^t, \tag{3.17}$$

$$\mathbf{v}^{t+1} = \mathbf{v}^t + \Delta t\mathbf{M}^{-1}\mathbf{f}(\mathbf{x}^{t+\frac{1}{2}}, \mathbf{v}^{t+\frac{1}{2}}), \tag{3.18}$$

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \Delta t\mathbf{v}^{t+\frac{1}{2}}. \tag{3.19}$$

Substituting Equation 3.16 into Equation 3.19 gives the formula:

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \Delta t\dot{\mathbf{x}}^t + \frac{\Delta t^2}{2!}\ddot{\mathbf{x}}^t + O(\Delta t^3) \quad ,$$

which is the same as the truncated Taylor series up to the second order. In other words, the second order Runge-Kutta integration achieves a third order accuracy. It requires twice the computational time of the explicit Euler scheme, as the force is evaluated twice in Equations 3.16 and 3.18. However, it is worth adopting this technique for applications where accuracy is needed, because doubling the costs gains four times the accuracy. Moreover, this improvement in accuracy makes simulations more stable. The explicit Euler integration can be seen as a first-order Runge-Kutta method. There are higher order Runge Kutta integration methods that provide yet higher accuracy and better stability. The fourth order Runge Kutta integration scheme that achieves fifth order accuracy is one the most popular integrators among explicit integration methods for physics simulation in computer graphics.

3.2.2 Implicit Methods

Although advanced explicit integration schemes improve the issue of stability, this problem is not totally solved. Another type of approach is implicit integration methods. The simplest

one is the implicit backward Euler method. Since instability results from the fact that explicit integrators blindly evaluate future state, the main strategy of implicit methods is to evaluate the force using the positions and velocities at the end of the time step, rather than using the current state as for explicit methods

$$\mathbf{v}^{t+1} = \mathbf{v}^t + \Delta t \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}^{t+1}, \mathbf{v}^{t+1}), \tag{3.20}$$

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \Delta t \mathbf{v}^{t+1}. \tag{3.21}$$

Compared to Equation 3.12 and 3.13, the position $\mathbf{x}^t$ and velocity $\mathbf{v}^t$ are replaced with $\mathbf{x}^{t+1}$ and $\mathbf{v}^{t+1}$. Clearly direct evaluating forces with unknown $\mathbf{x}^{t+1}$ and $\mathbf{v}^{t+1}$ is impossible. The first step is to regroup these two equations

$$\Delta\mathbf{v} = \mathbf{v}^{t+1} - \mathbf{v}^t = \Delta t \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}^{t+1}, \mathbf{v}^{t+1}), \tag{3.22}$$

$$\Delta\mathbf{x} = \mathbf{x}^{t+1} - \mathbf{x}^t = \Delta t \mathbf{v}^{t+1} = \Delta t(\mathbf{v}^t + \Delta\mathbf{v}). \tag{3.23}$$

and then to linearise the nonlinear system by applying a Taylor series expansion again but to force

$$\mathbf{f}(\mathbf{x}^{t+1}, \mathbf{v}^{t+1}) = \mathbf{f}^t + \mathbf{K}_\mathbf{x}\Delta\mathbf{x} + \mathbf{K}_\mathbf{v}\Delta\mathbf{v} \quad , \tag{3.24}$$

where $\mathbf{K}_\mathbf{x} = \partial\mathbf{f}/\partial\mathbf{x}$ and $\mathbf{K}_\mathbf{v} = \partial\mathbf{f}/\partial\mathbf{v}$ are Jacobians of force. They are $3N \times 3N$ matrices containing the derivatives of force with respect to positions and velocities, which are N^2 small 3×3 sub-matrices organised in a block fashion. For a given pair of particles i and j belonging to a triangle, the small derivative sub-matrix of the force acting on particle i with respect to the position of particle j is $\mathbf{K}_{\mathbf{x}_{ij}} = \partial\mathbf{f}_i/\partial\mathbf{x}_j \in \mathbb{R}^{3\times3}$, located in the ith row and jth column of matrices $\mathbf{K}_\mathbf{x}$. Since $\mathbf{K}_{\mathbf{x}_{ij}} = \mathbf{K}_{\mathbf{x}_{ji}}$, $\mathbf{K}_\mathbf{x}$ is symmetric. The definition is the same for $\mathbf{K}_\mathbf{v}$. Additionally, each particle is associated with a few number of other particles in both cases of in-plane and out-of-plane forces, hence $\mathbf{K}_\mathbf{x}$ and $\mathbf{K}_\mathbf{v}$ are sparse.

Substituting Equations 3.24 into 3.22 and regrouping Equation 3.22 and 3.23 gives

$$(\mathbf{M} - \Delta t\mathbf{K}_\mathbf{v} - \Delta t^2\mathbf{K}_\mathbf{x})\Delta\mathbf{v} = \Delta t(\mathbf{f}^t + \mathbf{K}_\mathbf{x}\mathbf{v}^t). \tag{3.25}$$

Letting

$$\mathbf{A} = (\mathbf{M} - \Delta t\mathbf{K}_\mathbf{v} - \Delta t^2\mathbf{K}_\mathbf{x}) \quad \text{and} \quad \mathbf{b} = \Delta t(\mathbf{f}^t + \mathbf{K}_\mathbf{x}\mathbf{v}^t)$$

gives

$$\mathbf{A}\Delta\mathbf{v} = \mathbf{b},$$

where matrix $\mathbf{A} \in \mathbb{R}^{3N\times 3N}$ is symmetric and $\mathbf{v}$ is a $3N$ dimensional vector. Then the problem is to solve $\mathbf{v}$ which can be efficiently done by a conjugate gradient solver. By having $\mathbf{v}$ the state at the end of the time step is trivially obtained.

Now how to compute matrices $\mathbf{K_x}$ and $\mathbf{K_v}$ is the remaining problem, which depends on the internal force model. For the spring based force model, sub-matrices $\mathbf{K}_{\mathbf{x}_{ij}}$ and $\mathbf{K}_{\mathbf{v}_{ij}}$ are a first derivative of force,

$$\begin{aligned}\mathbf{K}_{\mathbf{x}_{ij}} = \frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_j} &= \frac{\partial \mathbf{f}_i^s}{\partial \mathbf{x}_j} + \frac{\partial \mathbf{f}_i^d}{\partial \mathbf{x}_j} \\ &= k_s\left(\mathbf{I} - \frac{l}{|\mathbf{x}_j - \mathbf{x}_i|}\left(\mathbf{I} - \frac{(\mathbf{x}_j - \mathbf{x}_i)(\mathbf{x}_j - \mathbf{x}_i)^T}{(\mathbf{x}_j - \mathbf{x}_i)(\mathbf{x}_j - \mathbf{x}_i)}\right)\right) \\ &\quad - k_d\frac{(\mathbf{v}_j - \mathbf{v}_i)^T}{|\mathbf{x}_j - \mathbf{x}_i|}\left(\mathbf{I} - \frac{(\mathbf{x}_j - \mathbf{x}_i)(\mathbf{x}_j - \mathbf{x}_i)^T}{(\mathbf{x}_j - \mathbf{x}_i)(\mathbf{x}_j - \mathbf{x}_i)}\right)\end{aligned} \tag{3.26}$$

and

$$\mathbf{K}_{\mathbf{v}_{ij}} = \frac{\partial \mathbf{f}_i}{\partial \mathbf{v}_j} = k_d\left(\mathbf{I}\frac{\mathbf{x}_j - \mathbf{x}_i}{|\mathbf{x}_j - \mathbf{x}_i|}\right), \tag{3.27}$$

where $\mathbf{I}$ denotes the identity matrix. Entries of all pairs of particles that are not associated with each other are zeros.

For the constraint based force model, $\mathbf{K}_{\mathbf{x}_{ij}}$ and $\mathbf{K}_{\mathbf{v}_{ij}}$ are a second derivative of energy which depends on the constraint,

$$\mathbf{K}_{\mathbf{x}_{ij}} = \frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_j} = \text{-}k\left(\frac{\partial \mathbf{C}(\mathbf{x})}{\partial \mathbf{x}_i}\frac{\partial \mathbf{C}(\mathbf{x})}{\partial \mathbf{x}_j}^T + \frac{\partial^2 \mathbf{C}(\mathbf{x})}{\partial \mathbf{x}_i \partial \mathbf{x}_j}\mathbf{C}(\mathbf{x})\right), \quad \text{and} \tag{3.28}$$

$$\mathbf{K}_{\mathbf{v}_{ij}} = \frac{\partial \mathbf{f}_i}{\partial \mathbf{v}_j} = \text{-}k_d\frac{\partial \mathbf{C}(\mathbf{x})}{\partial \mathbf{x}_i}\frac{\partial}{\partial \mathbf{v}_j}\left(\frac{\partial \mathbf{C}(\mathbf{x})}{\partial \mathbf{x}_j}^T \dot{\mathbf{x}}\right) = \text{-}k_d\frac{\partial \mathbf{C}(\mathbf{x})}{\partial \mathbf{x}_i}\frac{\partial \mathbf{C}(\mathbf{x})}{\partial \mathbf{x}_j}^T. \tag{3.29}$$

Derivatives of different types of internal forces and damping forces are obtained by plugging constraints 3.6, 3.7 and 3.9 into these two equations. With this implicit integration method, stability can be achieved even if the time step size is very large or the system is very stiff. The implicit backward Euler method described in this section is first order. Higher order implicit integrators can be used to obtain higher accuracy, but with complexity and cost increases. It is noticeable that implicit integration can introduce artificial damping effects which is known as numerical dissipation, and the phenomenon that the larger the step size the more

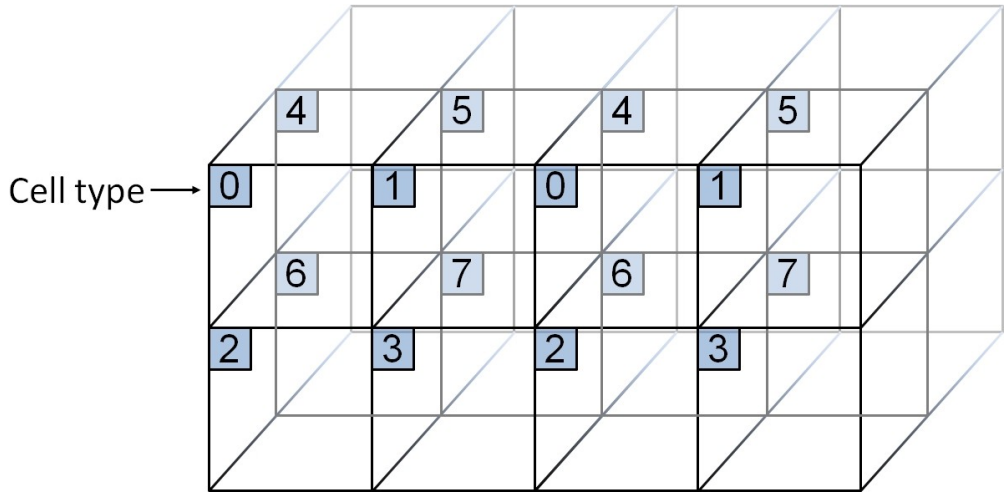


Figure 3.3: Cell types in the control bits scheme [Grand, 2007]

damping is generated is observed. Hence, Equation 3.8 and $\mathbf{K_v}$ are dropped strategically in some works using implicit integrators.

3.3 Collision Handling

Cloth objects in the real world interact with human bodies and other objects, and also easily fold, therefore complex collisions and contacts between cloth and external bodies or themselves are involved. This issue must be addressed physically and robustly for a realistic virtual cloth scenario. Moreover an efficient framework is necessary as the number of collisions and contacts involved can be huge.

Collision detection algorithms presented in Chapter 5 and 6 improve on the spatial subdivision technique using the GPU, therefore this section describes the basic and primary idea of a GPU-based spatial subdivision collision detection method. A robust treatment for collision response proposed by [Bridson et al., 2002] is described, while the friction component is replaced with an improved friction model discussed in Chapter 7.

3.3.1 Collision Detection

The spatial subdivision technique used in the broad phase is similar to [Grand, 2007]. The narrow phase responsible for elementary tests combines a number of techniques.

All triangles in a given space can potentially collide with each other. Clearly, performing collision tests for all triangle pairs, which is known as the brute-force method having a

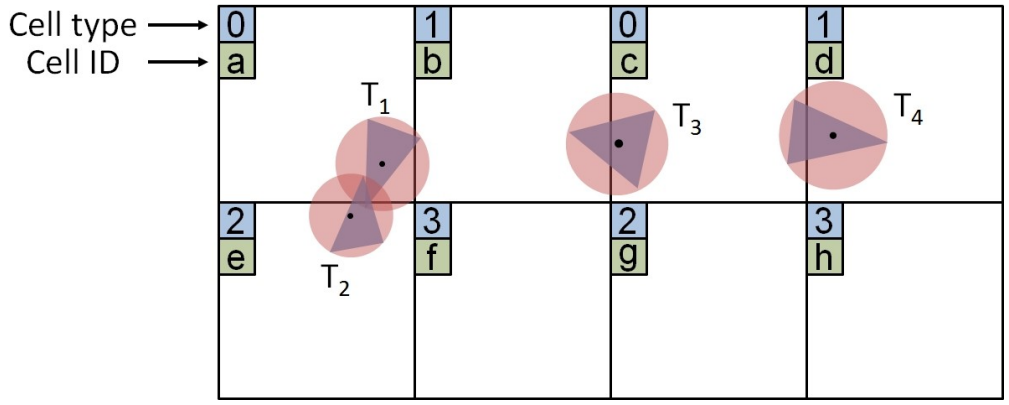


Figure 3.4: A 2D example of the control bits scheme to avoid duplicate tests due to triangle pairs assigned to more than one cell.

complexity of $O(n^2)$, is impractical when the number of objects is large. To decrease the complexity, space can be subdivided into a uniform grid, and each triangle assigned to cells that intersect the triangle's BV. Then a BV test is performed only for a pair of triangles that belong to the same cell, therefore BV tests for triangle pairs that are far from each other are avoided, giving an average complexity of $O(n)$.

However spacial subdivision gives rise to an issue that the same collision test for a pair of triangles may be performed multiple times if a triangle pair is assigned to more than one cell. To efficiently avoid such duplicate tests on the GPU, a control-bits scheme is used. In this scheme, a sphere is used as the BV. The first step is to set the cell size 1.5 times as large as the BV of the largest triangle, hence a triangle can be assigned to up to eight cells. Due to this feature, cells are classified into eight types in a fashion shown in Figure 3.3. For each triangle, a set of eleven control bits is used to indicate the type of its home cell where its centroid resides and types of cells intersected by the BV of the triangle: the first three bits specify the type number of the home cell, and each of the following eight bits indicate whether the BV intersects the corresponding cell. Then a filter test using control bits is performed for every triangle pair that belongs to the same cell and at least one of them has its centroid located in this cell. This test aims to find out whether one of the triangles' home cell type is both less than the type of the cell where this test occurs and among cell types that are common to both triangles. If this condition does not hold, a BV test is performed for this triangle pair.

For example, a 2D case is shown in Figure 3.4. For triangles T_1 and T_2 the BV test in cell a will be processed and the test in cell e will be skipped, because T_1's home cell type

is less than T_2's home cell type and they both intersect cell types 0 and 2. In another case, the BV test for the pair of T_3 and T_4 occurring in cell c can be skipped because they are too far apart to collide as the cell size is 1.5 times as large as the largest triangle's BV, which can also be examined by the filter test: T_4's home cell type is less than that of T_3 and both triangles share their home cell's type. It is unnecessary to perform the BV test for triangles T_1 and T_3 as well, because they are too far from each other as neither of their centroids are located in the only common cell b.

In the BV test, the R-triangle scheme [Curtis et al., 2008] is first used to remove duplicate vertex-triangle and edge-edge elementary tests due to the fact that a vertex and an edge can belong to multiple triangles, then BV culling based on vertices and edges is carried out to further reduce false positives.

The detection algorithm from here is referred to as the narrow phase. The output vertex-triangle and edge-edge pairs undergo elementary tests. In the case of discrete collision detection, a proximity test is performed to calculate intersection information at each discrete time step, which starts by checking if four vertices are close to a plane. For a given vertex l and a triangle with vertices i, j and k, the distance from the vertex to the triangle is first computed

$$d = |(\mathbf{x}_l - \mathbf{x}_k) \cdot \mathbf{n}|,$$

where $\mathbf{n}$ is the normal of the triangle. If d is less than a user defined cloth thickness h, the barycentric coordinates w_i, w_j and w_k of the projection of vertex l onto the plane containing the triangle is calculated by solving

$$\begin{bmatrix} (\mathbf{x}_i - \mathbf{x}_k) \cdot (\mathbf{x}_i - \mathbf{x}_k) & (\mathbf{x}_i - \mathbf{x}_k) \cdot (\mathbf{x}_j - \mathbf{x}_k) \\ (\mathbf{x}_i - \mathbf{x}_k) \cdot (\mathbf{x}_j - \mathbf{x}_k) & (\mathbf{x}_j - \mathbf{x}_k) \cdot (\mathbf{x}_j - \mathbf{x}_k) \end{bmatrix} \begin{bmatrix} w_i \\ w_j \end{bmatrix} = \begin{bmatrix} (\mathbf{x}_i - \mathbf{x}_k) \cdot (\mathbf{x}_l - \mathbf{x}_k) \\ (\mathbf{x}_j - \mathbf{x}_k) \cdot (\mathbf{x}_l - \mathbf{x}_k) \end{bmatrix}, \tag{3.30}$$

$$1 - w_i - w_j - w_k = 0. \tag{3.31}$$

Barycentric coordinates falling into the interval $[-\delta, 1 + \delta]$, where δ is h divided by the triangle's characteristic length, indicates that the vertex is close to the triangle and barycentric coordinates are output for repulsion force computation in the collision response stage.

Whether an edge with vertices i and j and another edge with vertices k and l are close depends on the distance between two closest points, one on each edge. If two edges are nearly parallel, the computation degrades to a one dimensional problem. Otherwise the following

equation needs to be solved for weights a and b:

$$\begin{bmatrix} (\mathbf{x}_j - \mathbf{x}_i) \cdot (\mathbf{x}_j - \mathbf{x}_i) & -(\mathbf{x}_j - \mathbf{x}_i) \cdot (\mathbf{x}_l - \mathbf{x}_k) \\ -(\mathbf{x}_j - \mathbf{x}_i) \cdot (\mathbf{x}_l - \mathbf{x}_k) & (\mathbf{x}_l - \mathbf{x}_k) \cdot (\mathbf{x}_l - \mathbf{x}_k) \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} (\mathbf{x}_j - \mathbf{x}_i) \cdot (\mathbf{x}_k - \mathbf{x}_i) \\ -(\mathbf{x}_l - \mathbf{x}_k) \cdot (\mathbf{x}_k - \mathbf{x}_i) \end{bmatrix}. \quad (3.32)$$

If $0 \leq a, b \leq 1$ the closest points are on edges. Otherwise the pair of points are clamped to ends of edges. The distance between two points is used to determine if two edges are close, if so operations similar to the vertex-triangle case are performed.

In the case of continuous collision detection, the path of four moving vertices during the entire time step has to be considered to test coplanarity, which is achieved by solving a cubic equation for t

$$((\mathbf{x}_j - \mathbf{x}_i) + t(\mathbf{v}_j - \mathbf{v}_i)) \times ((\mathbf{x}_k - \mathbf{x}_i) + t(\mathbf{v}_k - \mathbf{v}_i)) \cdot ((\mathbf{x}_l - \mathbf{x}_i) + t(\mathbf{v}_l - \mathbf{v}_i)) = 0. \quad (3.33)$$

Roots that do not fall into the interval $[0, \Delta t]$ are discarded. The positions of four vertices at each remaining time instant t are used to do proximity tests described above. If they are closer than a thickness (e.g. $h/1000$), they are registered as a collision and related contact information is sent to the collision response stage.

3.3.2 Collision Response

Once collision information is computed, the next step is to alter the velocity of vertices involved and thus prevent penetration. To achieve this, an impulse is computed for solving contacts or collisions found in both cases of discrete and continuous collision detection.

Impulses are computed for two colliding points which can be a vertex and an interior point of a triangle or two points on two edges each. The velocities of these two points are needed for calculating response. They are computed using barycentric coordinates using linear interpolation. The velocity of a point having barycentric coordinates w_i, w_j and w_k interior to a triangle is $w_i\mathbf{v}_i + w_j\mathbf{v}_j + w_k\mathbf{v}_k$ and the velocity of a point on an edge with the weight a is $(1 - a)\mathbf{v}_i + a\mathbf{v}_j$.

For contacts detected in the discrete time step, an elastic impulse is applied to separate the vertex and triangle or two edges that are close. This is explained as a compression of cloth fibers when two pieces of cloth are in contact. Hence it is reasonable to form the elastic impulse like a spring force that the more the overlap beyond the cloth thickness h the higher impulse is generated. However if the stiffness of the elastic impulse is set too large, a bouncing

effect will be introduced, which reduces the realism. Therefore the impulse is limited to a maximum that two pieces of cloth will not be propelled outside the overlap region in one time step, no matter what the stiffness is. This also helps pieces of cloth in contact to feel friction in subsequent time steps, leading to more realistic folds. The elastic impulse is based on the overlap depth which is computed as

$$d = h - (\mathbf{x}_l - w_i\mathbf{x}_i - w_j\mathbf{x}_j - w_k\mathbf{x}_k) \cdot \mathbf{n} \tag{3.34}$$

for the vertex-triangle case, where $\mathbf{n}$ is the triangle normal. The overlap depth of two edges is

$$d = h - (\mathbf{x}_k + b(\mathbf{x}_l - \mathbf{x}_k) - \mathbf{x}_i - a(\mathbf{x}_j - \mathbf{x}_i) \cdot \mathbf{n}, \tag{3.35}$$

where $\mathbf{n}$ is the direction from one of the two closest points to the other. An impulse is a force times a time interval, so the elastic impulse can be obtained:

$$I_r = \text{-}\min\left(\Delta t k_r d, m\left(\frac{0.1d}{\Delta t} - v_n\right)\right), \tag{3.36}$$

where v_n is the relative velocity between contacting points in the normal direction, and k_r is the elastic stiffness. As the equation shows, if v_n times the time step size Δt is greater than or equal to a fraction of the overlap ($0.1d$), no repulsion will be applied. To avoid problems with stiffness, the force is limited to reduce the overlap by a fraction of the cloth thickness ($0.1h$ in our work) at most during one time step.

Although applying a repulsion force at the discrete time step to separate pieces of cloth in contact can dramatically decrease the number of collisions in the current and subsequent time steps, it does not guarantee to remove all intersections. If a vertex and a triangle or two edges are moving too quickly towards each other, the elastic impulse may be not high enough to cancel out the relative velocity in the normal direction before they interpenetrate. Therefore an inelastic impulse $I_c = mv_n/2$ is used to stop the collision immediately. This impulse is applied when collisions are found in continuous collision detection.

3.3.3 Friction

Friction is indispensable in physically based simulation. It is separated into static and kinetic friction for motionless and sliding contacts respectively. The classic Coulomb model is

popular in computer graphics, which can be easily achieved by applying impulses as well:

$$I_f = |\mathbf{v}_t^{new} - \mathbf{v}_t| m \tag{3.37}$$

$$\mathbf{v}_t^{new} = \max\left(1 - \mu\frac{\Delta v_n}{|\mathbf{v}_t|}, 0\right)\mathbf{v}_t, \tag{3.38}$$

where $\mathbf{v}_t$ and $\mathbf{v}_t^{new}$ are relative tangential velocities before and after friction respectively. The second equation shows that if $\mu\Delta v_n$ is larger than $|\mathbf{v}_t|$, the relative tangential velocity is cancelled out, which is the static friction case, otherwise the relative tangential motion is slowed down due to kinetic friction.

The overall impulse is computed for two colliding points so far. It has to be propagated to the triangle corners or edge endpoints to update velocity. For the vertex-triangle case, the new velocities of vertices are

$$\bar{I} = \frac{2I}{1 + w_i^2 + w_j^2 + w_k^2}, \tag{3.39}$$

$$\begin{aligned}
\mathbf{v}_x^{new} &= \mathbf{v}_x + w_i(\bar{I}/m)\mathbf{n} \quad x = i, j, k, \\
\mathbf{v}_l^{new} &= \mathbf{v}_l - (\bar{I}/m)\mathbf{n}.
\end{aligned}$$

For the edge-edge case the computation is

$$\bar{I} = \frac{2I}{a^2 + (1-a)^2 + b^2 + (1-b)^2}, \tag{3.40}$$

$$\begin{aligned}
\mathbf{v}_i^{new} &= \mathbf{v}_i + (1-a)(\bar{I}/m)\mathbf{n}, \\
\mathbf{v}_j^{new} &= \mathbf{v}_j + a(\bar{I}/m)\mathbf{n}, \\
\mathbf{v}_k^{new} &= \mathbf{v}_k - (1-b)(\bar{I}/m)\mathbf{n}, \\
\mathbf{v}_l^{new} &= \mathbf{v}_l - b(\bar{I}/m)\mathbf{n}.
\end{aligned}$$

Velocities are updated together, which means that impulses acting on a vertex are summed up and divided by the number of interactions that it is involved in. Then the velocity is updated using equations described above with the average impulse. However, resolving collisions may introduce new collisions because the case of vertex-triangle or edge-edge are handled independently from each other, which is referred to as the problem of multiple collisions. Hence the operations of collision detection and applying impulses have to be performed iteratively until the cloth is collision free. Unfortunately [Provot, 1997] point out that this iterative scheme is not guaranteed to converge. To address this issue, a scheme called "rigid impact zones" is used. After a few iterations, vertices that are involved in new

introduced collisions are grouped together to form an "impact zone" which is treated as a rigid body. The impact zone grows until a collision free state is achieved, which is guaranteed to converge when the impact zone includes all vertices in the worst case.

The first step is to compute the initial mass center and the average velocity of the impact zone

$$\mathbf{x}_{CM} = \frac{\sum_i m\mathbf{x}}{\sum_i m}, \quad \mathbf{v}_{CM} = \frac{\sum_i m\mathbf{v}}{\sum_i m}$$

and its angular momentum with respect to the mass center

$$\mathbf{L} = \sum_i m(\mathbf{x}_i - \mathbf{x}_{CM}) \times (\mathbf{v}_i - \mathbf{v}_{CM}).$$

Its inertia tensor which is a 3×3 matrix is computed as

$$\mathbf{I} = \sum_i m(|\mathbf{x}_i - \mathbf{x}_{CM}|^2\delta - (\mathbf{x}_i - \mathbf{x}_{CM}) \otimes (\mathbf{x}_i - \mathbf{x}_{CM})).$$

To preserve angular momentum, the rigid body angular velocity is needed

$$\mathbf{v}_\omega = \mathbf{I}^{-1}\mathbf{L}.$$

Since the impact zone is treated as a rigid body, lengths and angles should be unchanged during the rotation. Therefore for a vertex i belonging to the impact zone the fixed and rotating components of the position should be considered:

$$\mathbf{x}_F = (\mathbf{x}_i - \mathbf{x}_{CM}) \cdot \frac{\mathbf{v}_\omega}{|\mathbf{v}_\omega|}\frac{\mathbf{v}_\omega}{|\mathbf{v}_\omega|},$$

$$\mathbf{x}_R = \mathbf{x}_i - \mathbf{x}_{CM} - \mathbf{x}_F.$$

The updated position is:

$$\mathbf{x}_{CM} + \Delta t\mathbf{v}_{CM} + \mathbf{x}_F + \cos(\Delta t|\mathbf{v}_\omega|)\mathbf{x}_R + \sin(\Delta t|\mathbf{v}_\omega|)\frac{\mathbf{v}_\omega}{|\mathbf{v}_\omega|} \times \mathbf{x}_R.$$

The new velocity can be obtained by subtracting the old position from the new position and then divide it by the time step size.

Chapter 4

Modelling Bending Behaviour Using Hysteresis

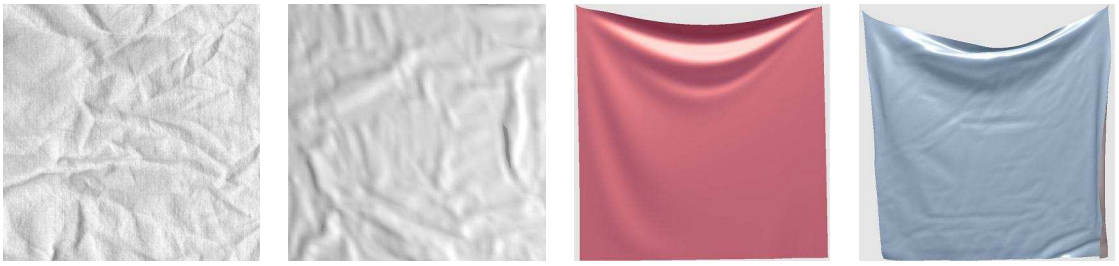
As discussed in Chapter 1, real cloth exhibits bending effects such as residual curvatures and permanent wrinkles. These are typically explained by bending plastic deformation due to internal friction in the fibre and yarn structure. Internal friction also gives rise to energy dissipation which significantly affects cloth dynamic behaviour. In textile research hysteresis is used to analyse these effects, and can be modelled using complex friction terms at the fabric geometric structure level. The hysteresis loop is central to the modelling and understanding of elastic and inelastic (plastic) behaviour, and is often measured as a physical characteristic to analyse and predict fabric behaviour. However, in cloth simulation in computer graphics the use of hysteresis to capture these effects to our knowledge had not been reported prior to our work. Previous approaches have typically used plasticity models for simulating plastic deformation. This chapter reports our investigation into the use of a simple mathematical approximation to an ideal hysteresis loop at a high level to capture the previously mentioned effects. Fatigue weakening effects during repeated flexural deformation are also considered based on the hysteresis model. Comparisons with previous bending models and plasticity methods are provided to point out differences and advantages. The method only requires little additional computation time.

4.1 Introduction

Establishing a good bending model is a key aspect for realistic cloth simulation involving wrinkling and folding. As an important textile property, bending behaviour shows high

nonlinearity which in textile research is described by a rheological model that contains a generalized bending rigidity and bending hysteresis [Owen, 1968; Chapman, 1975b; Bliman and Sorine, 1995a]. The latter physically describes nonlinear deformation and energy loss due to internal friction which also gives rise to plasticity. However, hysteresis has received relatively little attention in computer cloth simulation. Although many existing cloth modelling systems can generate good looking wrinkles and folds, physically based simulation of the bending plasticity relating to hysteresis has not been developed. Materials such as cloth, which exhibits energy loss and inelasticity during flexural deformation, may not return to the original configuration once external forces are removed, with some residual curvatures and permanent wrinkles. Such effects significantly influence realism and visual quality. Even in textile research fully accurate modelling bending hysteresis and plasticity has still been an open research topic. This chapter presents a model suited to computer graphics to simulate such features (Figure 4.1) with limited computational costs by using a simple mathematical approximation of bending hysteresis.

In bending few real materials are purely elastic. For most materials, small deformations are elastic up to the elastic limit, and then when exceeded plastic deformation occurs. The elastic limit is referred to as the transition point from the elastic to the plastic range. Once the deformation enters the plastic range, the deformation increases more rapidly than in the elastic range. For fabrics bending plasticity manifests as wrinkle and crease effects that may not be completely removed, which frequently happens to cloth materials that do not have good original appearance retention properties, such as cotton. This has been proven to be a consequence of internal friction in fibre and yarn structure. High friction between fibres may maintain their relative displacement if the elastic deformation limit is exceeded, and thus fabrics may not recover to its initial shape Zhao et al. [2009]. Internal friction also leads to nonlinearity and energy dissipation. The former occurs in the elastic range and is caused by a pre-sliding effect (due to stiction). The latter is observed no matter whether the elastic limit is exceeded or not, as bending requires more force and energy than unbending. These phenomena can be described by a hysteresis loop (Figure 4.3). The area of the loop is the dissipated energy. Quadratic curves are used to closely approximate the nonlinearity. Following [Anandjiwala and Leaf, 1991a], we take the transition point from nonlinearity (the quadratic region) to the rigidity slope (the linear region) as the elastic limit. For fabrics part of the dissipated energy results in fatigue weakening due to changes or jamming in the microstructure, leading to a decrease in internal friction and thus a decrease in hysteresis [Chapman, 1975a; Anandjiwala and Leaf, 1991a]. Therefore as a fabric is bent repeatedly,



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

Figure 4.1: Permanent wrinkles after large deformations in (a) real fabrics and (b) our simulation; Results from an (c) existing method (d) and our method

internal friction and hysteresis gradually decrease and finally disappear, leaving a saggy cloth with some permanent wrinkles and folds. Such permanent plastic deformation is hard to remove unless special conditions such as ironing or alkaline solutions are applied.

Previous methods to model plastic deformation typically use approaches where an elastic threshold is used to trigger plastic deformation [O'Brien et al., 2002; Kim et al., 2011]. These methods without considering hysteresis are able to simulate residual wrinkles, but the effect of energy dissipation which significantly affects the whole dynamic behaviour can not be fully captured, because there is no difference in force required to bend and unbend. In these methods, energy loss is achieved by empirically adjusting the damping component, while hysteresis is the way to describe and model energy dissipation in a physically accurate fashion. Furthermore, nonlinear bending behaviour in the elastic range [Clapp et al., 1990] is not simulated by the plasticity models as bending deformation is treated purely linearly. These plasticity methods can be extended to model fatigue weakening, but are limited to forming permanent wrinkles. The decrease in other aspects relating to hysteresis can not be captured. Therefore, motivated by such limitations a hysteresis based model for cloth bending behaviour is presented in this chapter.

Although our hysteresis approach is based on work from textile research, we do not directly adopt the methods into our cloth model for a few reasons. Firstly, the textile community focuses on investigating fabric behaviour at the microscopic fibre level using a complex fibre geometric model, which would drastically increase the computational complexity and cost, and is unlikely to be necessary for computer graphics purposes. Secondly, many parameters for measurements of real cloth materials, such as the interfibre force, space size between threads, change in yarn radius, etc., are required, but are unlikely to be important in graphics applications. Last but not least, their work focuses on the complete bending cycle, whilst in

computer graphics cloth usually only undergoes partial bending and recovery.

In this chapter, we present a physically based bending model suited to the widely used triangle mesh representation to compute bending forces using hysteresis at a high level without complex friction models. The primary idea is to use an ideal mathematical hysteresis loop that consists of two different curvature-moment curves for bending and recovery to look up bending and unbending forces, and which includes nonlinear bending stiffness. Furthermore, computing forces based on these two curves inherently incorporates energy dissipation due to internal friction, which alleviates high frequency oscillations, a common issue when small time steps are used in most cloth simulation methods [Thomaszewski et al., 2008]. Since plasticity arises from internal friction, residual curvatures are computed through hysteresis where the impact of internal friction on behaviour is included. Another contribution is to extend this model to consider fatigue weakening, turning residual wrinkles into permanent sets. Our model is not much more complicated than existing models, and experiments show that with a small extra computation time satisfactory bending hysteresis and plasticity can be obtained. This model can be easily integrated into previous cloth models by modifying the bending component. We show that this method can be implemented on GPUs to achieve high performance. For this work, we used the CUDA toolkit 4.1 on an NVIDIA GeForce 470 GTX GPU to conduct experiments.

The remainder of this chapter is organised as follows. We start by briefly reviewing previous work directly related to the bending behaviour and bending hysteresis effect in Section 4.2. The bending model is presented in Section 4.3, followed by the implementation and results in Sections 4.4 and 4.5 respectively. Finally, the conclusion is given Section 4.6.

4.2 Related Work

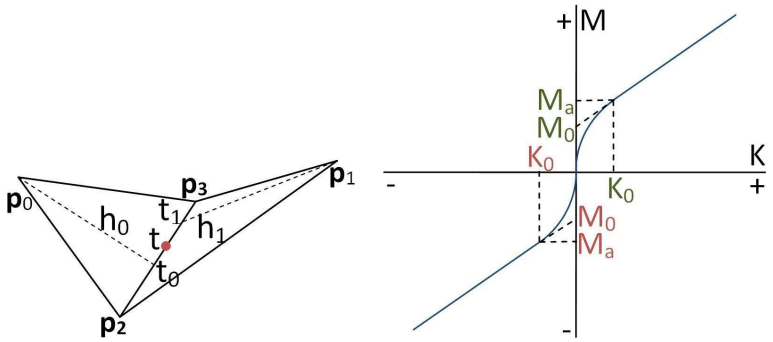
As described in Chapter 3, the bending force is formed by a spring to resist flexion in the mass-spring model. This kind of system is simple and fast, but unfortunately, it has the issue of inaccuracy in bending stiffness for small curvatures where a nonlinear phenomenon is exhibited. Baraff and Wiktin [Baraff and Witkin, 1998] present an elasticity-based model that derives the bending force from a bending energy function that takes account of the dihedral angle between two adjacent triangles. Bridson et al. [Bridson et al., 2003b] propose a simple and fast bending model that also uses dihedral angles. Grinspun et al. [Grinspun et al., 2003] present a flexural energy function that measures mean curvatures instead of Gaussian curvatures for membrane simulation. In their work, a creased and curled sheet of

paper is simulated by directly modifying the undeformed configuration, which can be seen as static approximations of plasticity like effects. However nonlinear bending behaviours are not mentioned in the approaches mentioned so far. Pabst et al. [Pabst et al., 2008] also use mean curvatures to compute bending force. This approach is based on the use of measured data describing the curvature-moment relationship presented by Clapp et al. [Clapp et al., 1990] from the textile community. These measured data closely approximates the nonlinear fabric bending stiffness during the bending process. Our bending model is an improvement on their approach. The systems mentioned so far do not consider hysteresis and plasticity of fibres.

The earliest plasticity model in graphics is in [Terzopoulos and Fleischer, 1988] where objects experience plastic deformation when the applied force exceeds an elastic limit force. Hysteresis for computer cloth simulation is first discussed by Breen et al. [Breen et al., 1992]. In order to simulate realistic cloth draping, they use a system called Kawabata to measure the hysteresis during a cycle of force loading and unloading on a piece of real cloth, but this work is designed to compute the static draping shape and do not produce cloth animation showing dynamic behaviour. Therefore, only the basis of fabric hysteresis has been discussed. Ngoc and Boivin [Ngoc and Boivin, 2004] use relatively complex friction terms for parameters identification and thus to be able to accurately reproduce nonlinear behaviours including hysteresis in cloth modelling. However quite high computational costs are required as a large number of iterations (up to hundreds) is required. Plasticity effects are not modelled in their work. Volino et al. [Volino et al., 2009] simulate tensile behaviour of cloth materials using nonlinear Green-Lagrange tensors. Their work discusses tensile plasticity along with hysteresis, and can be extended to model plasticity through a proper processing of the strain-stress behaviour. Kim et al. [Kim et al., 2011] generate residual changes in curvature by simply treating this feature as energy loss over time without a firm physical foundation and explanation to how and why the energy loss occurs. Hysteresis is not used in the computation and ten empirical parameters are required. Outside cloth simulation, O'Brien et al. [O'Brien et al., 2002] introduce tensile plasticity into solid simulation to model ductile objects by using an elastic strain and a plastic strain, but hysteresis is not considered.

4.3 Bending Force Model

The bending force model described in this section is different from models presented in Chapter 3.



(a) A bending element (b) Curvature-moment curve

Figure 4.2: (a) A bending element formed by two adjacent triangles; (b) The curvature-moment relationship. If the curvature $K > 0$, c_1 is a negative value and M_0, M_a and K_0 (green) are positive. Otherwise, c_1 is positive, and M_0, M_a and K_0 (red) are negative.

Since the nonlinearity of hysteresis can be expensive and impractical to handle implicitly, we adopt a semi-implicit scheme [Schroeder et al., 2011]. We use explicit integration on bending forces and implicit integration on tensile and damping forces. To ensure stability and obtain efficiency, a small and a large time step, h_e and h_i, are used for explicit and implicit integration respectively. We use $h_i = \alpha h_e$, where α is a positive integer. In fact, treating bending forces explicitly is not a limiting factor for stability, as bending forces are much weaker than tensile forces in cloth simulation [Thomaszewski et al., 2008].

4.3.1 Bending Force

We build the hysteresis bending model on the work presented by Pabst et al. [2008], by using different forces for bending and unbending. Their work is briefly presented here for readers' convenience. The model relies on the angular expression. By having the angle between two adjacent triangles (Figure 4.2(a)), it is able to use measured data of curvature-moment relationship (Figure 4.2(b)) to compute the bending force. Such data is acquired by Clapp et al. [1990] who present a method in textile research to indirectly measure the curvature and moment during a cantilever beam bending test. One of the most important advantages of their measurement is that both nonlinear and linear bending behaviours are present. Therefore a good accuracy can be attained.

The mean curvature K is obtained by considering the current dihedral angle θ, the initial

dihedral angle θ_0 and the average height $\bar{h}$ of two adjacent triangles in the undeformed state:

$$K = \frac{\theta - \theta_0}{\bar{h}} \tag{4.1}$$

The relationship between curvature K and moment M is

$$M = \begin{cases} b_1 K + c_1 K^2 & \text{if } |K| \leq |K_0| \\ M_0 + b_2 K & \text{if } |K| > |K_0| \end{cases} \tag{4.2}$$

where K_0 and M_0 are the transition curvature and frictional moment respectively, and computed as

$$K_0 = \frac{b_2 - b_1}{2c_1}$$

$$M_0 = (b_1 - b_2)K_0 + c_1 {K_0}^2$$

where b_1, b_2 ($b_1 \leq b_2$) and c_1 are constants for defining the bending stiffness. If K is negative the value of c_1 should change its sign. The bending stiffness B can be seen as a continuous function of the gradient of Equation 4.2:

$$B(K) = \frac{dM}{dK} \tag{4.3}$$

The curvature-moment relationship is illustrated in Figure 4.2(b). When the curvature is small, cloth displays nonlinearity and bending stiffness decreases as the bending curvature increases. The quadratic behaviour arises from the pre-sliding effect where friction takes a small displacement to build from zero to the maximum stiction force [Altpeter, 1999]. In cases of bending deformation, the displacement is replaced with a curvature (K_0 in this work). However the detailed study of how hysteresis is characterised by the pre-sliding effect is beyond the scope of this work. If the curvature is beyond K_0 ($M > M_a$), the frictional moment M_0 is overcome indicating that the frictional restraint is conquered (internal friction reaches the maximum stiction force). Then bending behaviours are represented by the linear part. The bending stiffness stops decreasing and has a constant value of b_2.

Similar to [Pabst et al., 2008] but with some small corrections, having computed the bending moment M using Equation 4.2, the bending force for two adjacent triangles, forming a bending element as shown in Figure 4.2(a) can be now obtained. We start by calculating

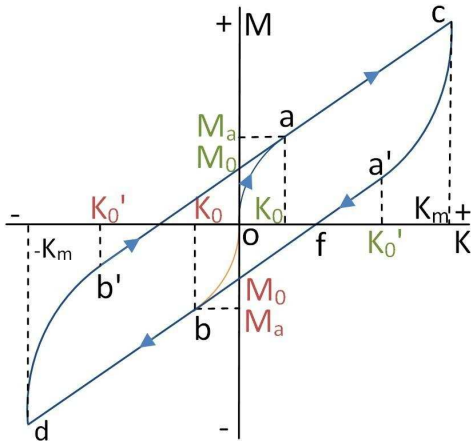


Figure 4.3: A typical bending hysteresis loop for woven fabrics.

barycentric coordinates t, t_0 and t_1 with respect to the common edge $\mathbf{e} = \mathbf{p}_3 - \mathbf{p}_2$:

$$t_i = \frac{\mathbf{e} \cdot (\mathbf{p}_i - \mathbf{p}_2)}{\mathbf{e} \cdot \mathbf{e}} \quad i \in \{0, 1\}$$

$$t = t_0 + (t_1 - t_0)\frac{h_0}{h_0 + h_1}$$

where h_0 and h_1 are the height of two triangles in the current state. The barycentric coordinates are used to distribute the bending force exerted on four vertices with the preservation of translational and rotational momentum:

$$\mathbf{f}_0 = \frac{1}{2}\frac{M}{|\mathbf{l}_0|}\frac{\mathbf{e} \times \mathbf{l}_0}{|\mathbf{e} \times \mathbf{l}_0|}$$

$$\mathbf{f}_1 = \frac{1}{2}\frac{M}{|\mathbf{l}_1|}\frac{\mathbf{l}_1 \times \mathbf{e}}{|\mathbf{e} \times \mathbf{l}_1|}$$

$$\mathbf{f}_2 = -(\mathbf{f}_0 t + \mathbf{f}_1(1 - t))$$

$$\mathbf{f}_3 = -(\mathbf{f}_0(1 - t) + \mathbf{f}_1 t)$$

$$\mathbf{l}_i = \mathbf{p}_i - (\mathbf{p}_2 + t_i \mathbf{e}) \quad i \in \{0, 1\}$$

4.3.2 Bending Hysteresis Loop

Internal friction plays a major role in hysteresis. Any slight movement in the fibre and yarn structure can create interfiber friction, causing energy dissipation during the cycle of force loading and unloading. The relationship between moment and curvature for such bending deformation is depicted in Figure 4.3 as a typical hysteresis loop [Owen, 1968; Chapman, 1975b]. Our bending model is based on this loop.

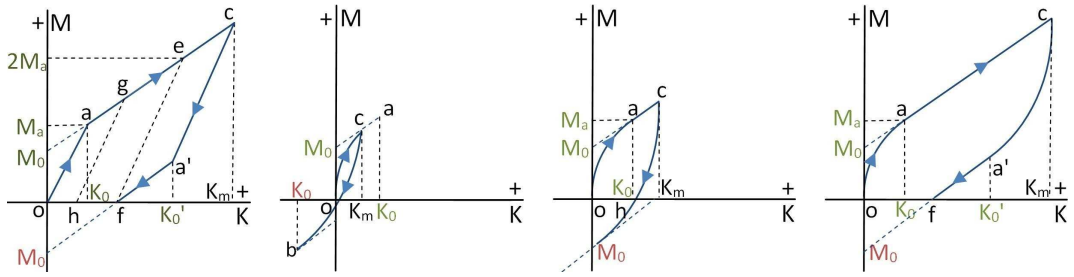
The theoretical loop is used to describe the following bending behaviours in textile research. At first, let the undeformed fabrics be bent to reach curvature K_m by applying an increasing external force. Next, the curvature change is reversed by gradually reducing the external force to recover deformation, and continuously bend the fabrics to reach curvature -K_m by increasing the force in the opposite direction. Then, the behaviour is repeated by reversing the direction again.

Compared to Figure 4.2(b), the most important difference is that the moment for bending and unbending are along different curves, implicitly introducing the energy dissipation through the loop area, which gives rise to a dampening effect, one consequence of which is to reduce fast oscillation behaviours. Similar to the curve oa, quadratic curves ca' and db' are formed to model nonlinear behaviour resulting from internal friction. These two curves describe the process that when the curvature change is reversed stiction drops to zero and then increases to maximum in the opposite direction. This is why the moment starts with a quadratic curve. In our work the stiction maximum is independent of the bending direction, which is true for most real cloth materials, so that the curvature required to complete the friction change from o to a equals that from o to b, and that from c to a' and from d to b' are twice K_0 ($K_m - {K_0}' = 2K_0$). The same happens to the moment change. Note that the curvature-moment relationship is C^1 continuous at a, a', b and b'.

The bending model presented in Section 4.3.1 is easily extended to model bending and recovery differently like the hysteresis loop shown in Figure 4.3. According to features discussed in the previous paragraph, we are able to construct the equation to describe curves ca' and db'. By translating them to the origin, the curves can be written as:

$$M = \beta_1 K + \gamma_1 K^2 \tag{4.4}$$

Since two ends of the quadratic curve are known, and curves are C^1 continuous on where the



(a) A bilinear model (b) $|M| \leq |M_a|$ (c) $|M_a| < |M| \leq 2|M_a|$ (d) $|M| > 2|M_a|$

Figure 4.4: (a) A bilinear bending hysteresis loop for woven fabrics. (b), (c), and (d) Three different recovery cases.

quadratic and linear curves join, the coefficients can be easily computed:

$$\begin{aligned} \beta_1 &= b_2 + \frac{2M_0}{K_0} \\ \gamma_1 &= \frac{\sigma_1 M_0}{2K_0^2} \end{aligned} \tag{4.5}$$

where σ_1 is set to -1 and 1 for db' and ca' respectively. However this loop represents the moment change for deformations of cloth undergoing large bending and unbending, but in simulation cloth may just undergo small flexural deformations, or get partially recovered from bending and then be bent again, as reversed curvature change can occur at any time. Hence quadratic curves can be different from $ca'f$ and db'. Computation of coefficients for these quadratic curves will be discussed later.

4.3.3 Residual Curvature

In order to generate plastic deformation, residual curvatures in large flexural deformation have to be computed. It is necessary to understand how internal friction affects the moment change in real woven fabrics, which has been studied in detail by Anandjiwala and Leaf [Anandjiwala and Leaf, 1991a] using a bilinear model (Figure 4.4(a)). In their work, the frictional restraint is approximately described by a line oa, and the line ac and $a'f$ represent the generalized linear bending stiffness b_2 just like ours.

There are three different situations for recovery (Figure 4.4(a)), depending on the bending moment M or the curvature K when recovery starts [Anandjiwala and Leaf, 1991a]. The following analysis is based on a test that fabrics are undergoing a fully recovered flexural deformation. In the first case of the curvature k being smaller than K_0 ($|M| \leq |M_a|$), the

moment will decrease along the line oa in the recovery so that the bending deformation will be completely reversible. If the frictional moment has been overcome and the bending moment is less than $2|M_a|$ ($|M_a| < |M| \leq 2|M_a|$), the recovery will be on the line gh having the same gradient of oa. For the last situation ($|M| > 2|M_a|$), the bending moment reduces along a line ca' parallel to oa and another line $a'f$ having the gradient b_2 successively. In the last two cases a residual curvature change will be introduced, because in fact when external forces are totally removed, a small amount of interfibre forces are still left resulting from internal dry frictions, leading to residual bending moment. It means that once the elastic limit is overcome plastic behaviours occur. Point a is regarded as the elastic limit.

Instead of computing the residual bending moment at a low level, we obtain residual curvature changes by the value of λK_r, where K_r is the curvature value at the intercept between the recovery curve and curvature axis, and $\lambda \in [0, 1]$ a parameter controlling the change rate. For example setting λ to 1 indicates that when the moment reaches zero the relative displacement between fibres is fully held by stiction. If it is set to 0 no plasticity occurs. This approximation works well at the simulation level and does not come at the expense of visual fidelity.

Similar to [Anandjiwala and Leaf, 1991a], in our model recovery behaviour is subdivided into three types. In the first one (Figure 4.4(b)), the moment decreases along a quadratic curve ending at point b. Since the frictional moment has not been overcome, no residual curvature change will be introduced. As discussed in the previous section, curvature reversal can occur when cloth undergoes small flexural deformations before the elastic limit has been reached, resulting in the recovery curve cb being different from ca' (Figure 4.3). For the second and third situations (Figure 4.4(c) and 4.4(d)), the recovery will be on curves ch and $ca'f$ respectively, and thus a residual curvature λK_r will be created. In the second situation the intercept h is on the quadratic curve, while the intercept f in the third case is on the linear curve. Because of the residual curvature change, the initial angle θ_0 in Equation 4.1 is replaced by the residual angle θ_r (the initial value of θ_r is equal to θ_0). The cases we have analysed so far are deformations with full recovery. However, it will be further complicated if the deformation is not fully unloaded (the moment does not reach zero), which often happens in real life cases. For example, a bending element is first bent to have a curvature K_m and a moment M_m, followed by just a small recovery reducing the curvature to K and the moment to M then bent again. In such a case, the value of the introduced residual curvature also relies on how complete the recovery was in the last loading-unloading cycle. Hence, how much the frictional moment has been overcome during the last recovery process decides the

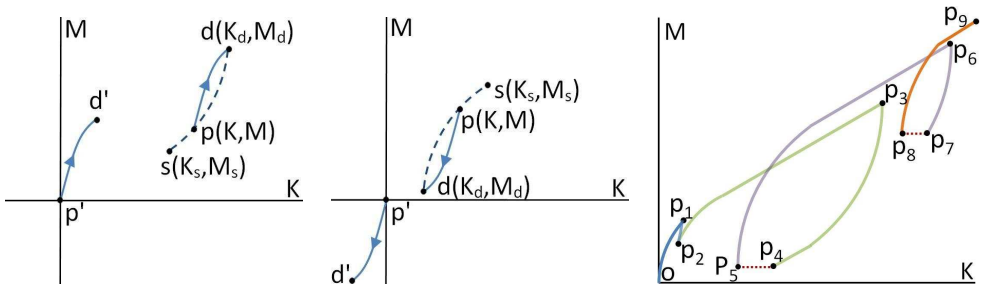


Figure 4.5: The solid curve pd is used for increase and decreasing moment in (a) and (b) respectively. Curve $p'd'$ is curve pd after a translation to the origin of the coordinate. $|K_d - K_s| = 2K_0$ and $|M_d - M_s| = 2M_a$. (c) The moment change for the first four sequential cycles is shown. op_1p_2, $p_2p_3p_4$, $p_5p_6p_7$ and p_8p_9 are curves for four cycles respectively. Disconnections (red dotted lines) are caused by residual curvature changes.

value of θ_r (superscripts indicate the cycle number):

$$\theta_r{}^{n+1} = \theta_r{}^{n} + \xi\lambda K_r\bar{h} \tag{4.6}$$

where $\xi \in [0, 1]$ is the ratio of how much the frictional moment has overcome and computed as

$$\xi = min\Big(1, \frac{M_m - M}{2M_a}\Big) \tag{4.7}$$

The residual angle is updated when each recovery ends if the frictional moment has been overcome. For this chapter, a cycle is defined as bending followed by recovery (unbending), even if only partial. The transition from recovery to bending or the bending element being completely recovered to the zero moment state indicates the end of a cycle.

4.3.4 Moment Change in A Practical Simulation System

As mentioned earlier, the direction of the flexural deformation can be reversed at any time. If it happens at points on a linear curve, the curve having the same shape with ca' or db (shown in Figure 4.3) is computed for the moment change. If the reversion point is on a quadratic curve, maximum stiction has not been reached so that a curve shorter than ca' or db will be used for the moment change. These curves can be represented by the solid quadratic curve pd in Figure 4.5. Point p can be anywhere on the dashed curve (if the endpoint p is on the

point s, pd is the same as ca' or db in Figure 4.3). Equation 4.5 is replaced by:

$$\begin{aligned} \beta_1 &= b_2 + \frac{2(2M_0\sigma_1 + b_2K_p - M_p)}{2K_0\sigma_1 - K_p} \\ \gamma_1 &= \frac{\text{-}2M_0\sigma_1 - b_2K_p + M_p}{(2K_0\sigma_1 - K_p)^2} \end{aligned} \tag{4.8}$$

where $K_p = K - K_s$, $M_p = M - M_s$, $\sigma_1 = 1$ when the quadratic curve is used for increasing moment (Figure 4.5(a)), and σ_1 = -1 when the quadratic curve is used for decreasing moment (Figure 4.5(b)). The moment can now be obtained by plugging β_1 and γ_1 to Equation 4.4.

Therefore, the curvature-moment relationship for sequential cycles in a practical simulation system could be like Figure 4.5(c). Note that disconnections p_4p_5, p_7p_8 (red dotted line) result from a residual curvature change introduced by the previous cycle, and causes a translation of the loop. The translation does not need to be computed explicitly as it is reflected in the model once the rest curvature is updated.

4.3.5 Repeated Wrinkling and Folding

One phenomenon observed during plastic deformation is that part of dissipated energy is consumed for fatigue weakening. In many situations some areas in cloth tend to exhibit wrinkles and creases more than other areas. For example, areas behind knees and elbows have more wrinkles and higher residual curvatures, as these parts are more likely to undergo larger flexural deformations repeatedly. In fact, once these residual wrinkles are created, during a repeated flexural deformation they will show a weaker bending rigidity than flat areas and thus more likely to be bent. This is why high residual curvatures are easily generated along existing crease lines.

This has been observed and discussed in [Chapman, 1975a; Anandjiwala and Leaf, 1991a] by analysing behaviour of real fabrics on which a number of cycles of loading and unloading are applied. Results show that at the end of each cycle, wrinkles along the same lines become more pronounced because additional residual curvature is introduced. However, the size of added residual curvature reduces gradually as the cycle number increases, and finally reaches zero towards infinity. As a result, bending hysteresis disappears and a permanent plastic deformation takes place in the fabric.

According to an analysis by Anandjiwala and Leaf [Anandjiwala and Leaf, 1991a], internal friction is not only the reason why residual curvature changes are created, but it also is a contributing factor to the decrease in bending hysteresis. As the cycle number increases,

the residual bending moment increases and finally is larger than the transitional moment M_a. During this process, at the beginning of bending in each cycle the bending stiffness is smaller than the previous cycle. Subsequently, fibres do not experience internal friction and loading (bending behaviour) does not need to conquer interfibre frictional moment any more, as friction between fibres has vanished because of jamming. The hysteresis loop becomes a straight line with a gradient of b_2, describing the generalized bending stiffness only. Hence, the bending rigidity in wrinkles is much weaker than that in flat areas. Without the presence of internal friction, new bending deformation can be completely recovered to the permanent plastic shape when external forces are released, because residual curvatures introduced before hysteresis disappears become permanent.

We propose a simple scheme that reduces the frictional moment M_0 at the end of each cycle to simulate this feature (superscripts indicate the cycle number):

$$M_0{}^{n+1} = (1-\mu)M_0{}^{n} \tag{4.9}$$

This is easy to achieve by adjusting b_1 to be closer to b_2, therefore we have

$$b_1{}^{n+1} = b_2 - \sqrt{1-\mu}(b_2 - b_1{}^{n}) \tag{4.10}$$

where μ is set between 0 and 1, and thought of as a parameter controlling the rate of the decrease in hysteresis effect. For example when $\mu = 0$ the hysteresis effect will never decrease, while by choosing a value close to 1 we can model materials, such as paper, that form permanent wrinkles quickly. According to the limit laws,

$$\lim_{n\to\infty} b_1{}^{n} = b_2$$

holds. This indicates that the loop degrades to a line and that hysteresis no longer exists ($M_0 = 0, K_0 = 0$), becoming just like many existing methods where bending forces are linear. The impact on the curvature-moment relationship is presented in Figure 4.6. As the area of the loop decreases, the contribution of frictional restraint to bending stiffness reduces. According to Equation 4.3, the initial flexural stiffness gradually decreases from b_1 to b_2.

However, as mentioned earlier the deformation is usually not completely recovered. How much the frictional moment has overcome during the last recovery should be considered.

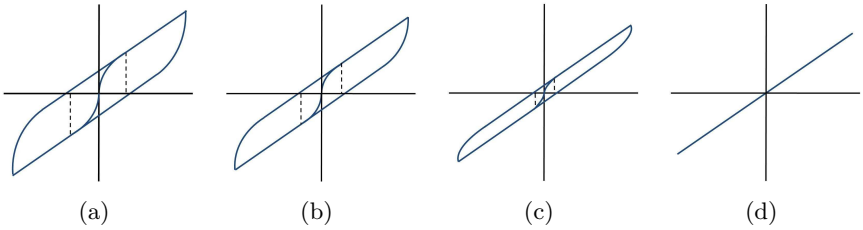


Figure 4.6: The change in bending hysteresis during multiple cycles: the hysteresis effect decreases as the cycle number increases, and finally completely disappears.

Equation 4.9 and 4.10 become:

$$M_0{}^{n+1} = (1 - \mu\xi)M_0{}^n \tag{4.11}$$

$$b_1{}^{n+1} = b_2 - \sqrt{1 - \mu\xi}(b_2 - b_1{}^n) \tag{4.12}$$

where ξ is computed using Equation 4.7. Therefore b_1 is not constant any more. Equation 4.12 is used to update b_1 when the recovery behaviour ends, if the frictional restraint is overcome in bending.

4.4 Implementation

This method can be integrated with the majority of existing cloth (and membrane) models that use triangle meshes, by modifying the bending model. The values of constants b_1, b_2 and c_1 determine the hysteresis effect and are chosen based on experimental bending properties on several different types of real cloth materials [Owen, 1968; Anandjiwala and Leaf, 1991b]. A better parametrization of the hysteresis controls may be possible, which we suggest as future work. The initial angle θ_0 can be used to initialize the original neutral object shape before simulation takes place. This is important for cloth simulation as real garments have intrinsic curvatures in many areas, such as shirt collars and pleats on skirt.

The energy-based in-plane force model is used to simulate stretching and shearing behaviours (Equations 3.4, 3.6 and 3.7). We adopt the implicit backward Euler scheme as the integrator to evolve tensile and damping forces (Equations 3.25, 3.28 and 3.29). The linear system generated by the implicit integration method is solved by using a preconditioned conjugate gradient (PCG) solver [Shewchuk, 1994]. A GPU-based virtual triangle subdivision scheme with a uniform grid is employed to handle collision detection, which will be presented in Chapter 5. For collision response a robust treatment [Bridson et al., 2002] (described in

Chapter 3) is used.

We use GPU computing to obtain higher performance by taking the advantage of the parallel GPU architecture. The bending model exhibits very high data parallelism, as the computation of bending forces for each bending element can be performed independently and the number of bending elements can be large. Therefore, the bending force computation is treated as a single kernel and each thread handles a bending element. The code is written in C++ using CUDA 4.1 with performance tests run on a machine with a 3.0GHz dual-core CPU and an NVIDIA GeForce 470 GTX GPU.

4.5 Results

We have built a range of scenarios to evaluate our method. We first provide and analyse curvature-moment plots of real data and data from our model. Then we compare outputs of our simulation to results generated by a previous bending model [Pabst et al., 2008] with respect to plastic deformation, by using three different scenarios: a twisted cloth sheet, a squished cloth bunny and a character animation. A similar effect for plasticity can be generated by previous plasticity models. Therefore, two more experiments are designed to demonstrate differences between our hysteresis model and existing plasticity models, including a test of hitting a cloth and an animation of a walking woman. Next, we perform a fatigue weakening experiment where a piece of cloth is repeatedly clamped by two parallel planes. For a single bending element, the data including residual curvatures, initial bending stiffness and changes in hysteresis is presented and analysed. Finally, a performance analysis is given. In order to exhibit plasticity effects within a short animation, values of λ and μ are relatively large in our experiments. In a longer simulation much smaller values would be used.

4.5.1 Curvature-moment Plots

Figure 4.7(a) shows the experimentally measured curvature-moment data for a fully loaded-unloaded bending test on a real acrylic twill weave fabric [Ngoc and Boivin, 2004]. The data was measured using the Kawabata Evaluation System (KES). Data obtained by performing the same test on a bending element using our method is given in Figure 4.7(b) (with the plasticity effect turned off), which shows that the hysteresis model reproduces the nonlinearity presented in real fabrics. It is noticeable that curvature-moment curves in our model are smoother than curves in real fabrics. This is not only because our method uses simple ideal

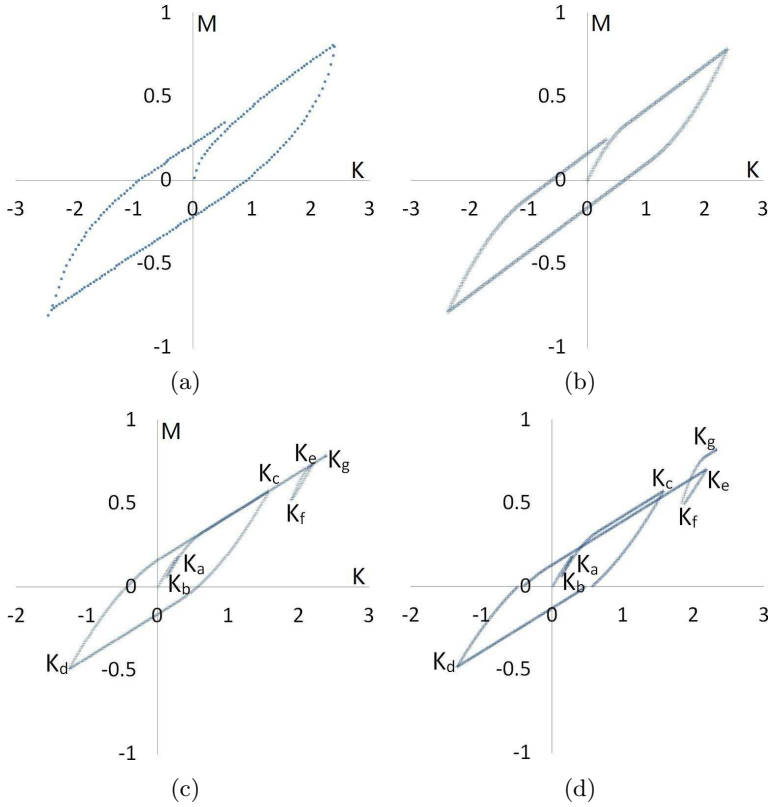
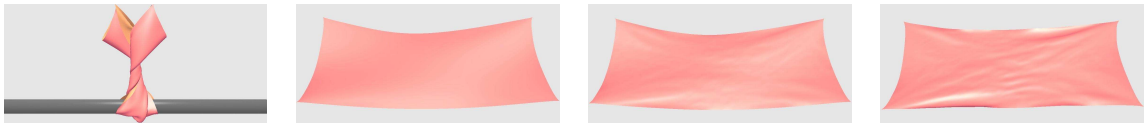


Figure 4.7: (a) Experimental data of a fully loaded-unloaded bending test on a real acrylic twill weave fabric from KES. (b) Data of the same test in (a) using our method. (c) Data of several sequential cycles of partial loaded-unloaded bending deformation using our method with plasticity effect turned off. The bending order is $0, K_a, K_b, K_c, K_d, K_e, K_f$ *and* K_g*. (d) Data of the same test in (c) but with plasticity effect turned on.* ($b_1 = 0.76, b_2 = 0.26$, c_1 = -0.38 *and* $\lambda = 0.1$*)*

quadratic and linear curves for moment changes, but also due to the fact that real fabrics exhibit a nonhomogeneous nature.

To further verify the method, a test, similar to the case discussed in Section 4.3.4, of several sequential cycles of partial loaded-unloaded bending deformation is performed. The curvature-moment data of this experiment with the plasticity effect turned off and on are shown in Figures 4.7(c) and 4.7(d) respectively. For applications that only require the ability of modelling nonlinearity, Figures 4.7(b) and 4.7(c) demonstrate that the method works well in practice. As expected, Figure 4.7(d) shows similar behaviour to that described in Section 4.3.4. Disconnections of curves between the transition from recovery to bending result from residual curvature changes.



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

Figure 4.8: (a) A cloth is twisted by a rotating rod, then allowed to recover. The difference in results between (b) an existing method and our bending hysteresis model with (c) low and (d) high plasticity are shown by spreading out the cloth.

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

Figure 4.9: (a) A cloth bunny is squished by a shrinking box, then allowed to recover. The difference in results between (b) an existing method and our bending hysteresis model with (c) low and (d) high plasticity are shown. Crumpling and dimples in (b) are formed by gravity.

4.5.2 Plasticity

We evaluate the plasticity effect in three scenarios. In the first case, a piece of cloth having four fixed corners is twisted by rotating a rod and then allowed to recover by rotating the rod in the opposite direction. In another case, a cloth bunny is first squished by a shrinking box, followed by retraction of the box to let the bunny recover. The differences in behaviour which result from our bending model and an approach without plasticity for these two scenarios are shown in Figures 4.8 and 4.9. After the steady state is reached, without plasticity the cloth, as expected, returns to its original bending configuration, with remaining bending deformation formed by external forces (i.e. gravity), while some residual wrinkles and folds are generated in our simulation. For experiments of low and high plasticity, λ are set to 0.08 and 0.25 respectively. Values of $b_1 = 4.64, b_2 = 1.40$, c_1 = -2.15 and $\mu = 0.02$ are used for both.

The character scene in which a dressed human model sits down and then stands up demonstrates that the bending model can be used to generate residual and permanent wrinkles for garments undergoing movement. This is shown in Figure 4.10, where our method yields different results especially in the groin region and in areas behind elbows and knees where large repeated flexural deformations occur, while the existing method without plasticity ends up with much smoother and simpler wrinkle shapes.

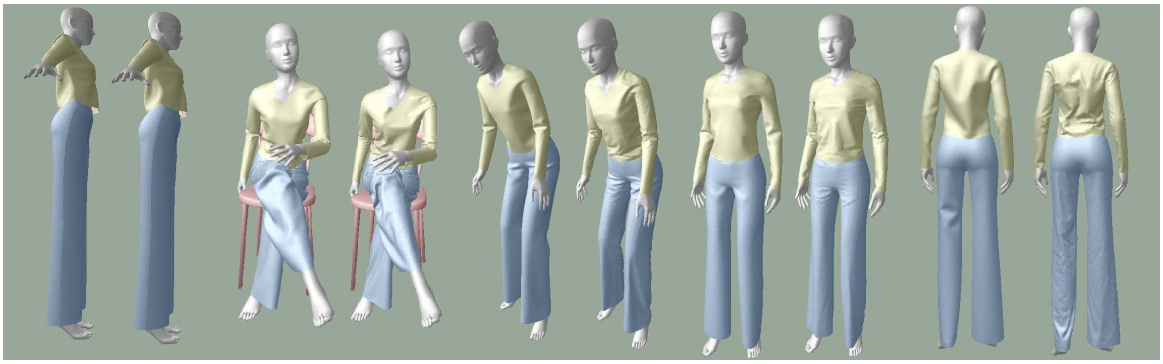
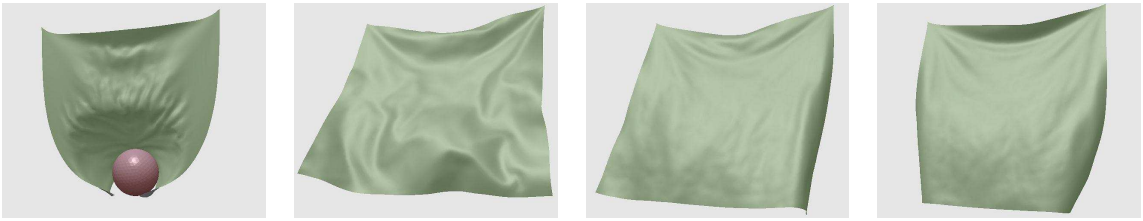


Figure 4.10: A dressed human character sitting down and then standing up, simulated by an existing method (left) and our method (right). In this test, coefficients are set to easily form plasticity effects ($b_1 = 3.4, b_2 = 2.85$, c_1 = -3.65, $\lambda = 0.3$ and $\mu = 0.05$).

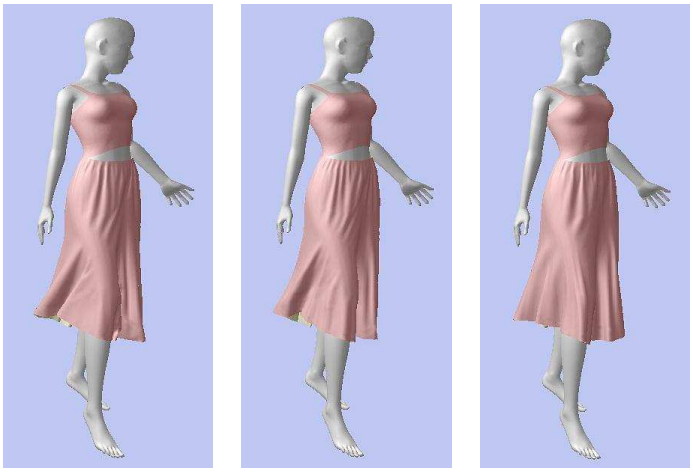


(a) (b) (c) (d)

Figure 4.11: (a) A fast sphere hits a piece of cloth. Results of (b) a method without plasticity and hysteresis, (c) plasticity method and (d) our hysteresis model on frame 230. ($b_1 = 3.4, b_2 = 2.0$, c_1 = -2.23, $\lambda = 0.2$ and $\mu = 0.02$)

4.5.3 Comparison with Previous Plasticity Models

Purely elastic deformation stores energy and releases it in recovery. In contrast plastic deformation absorbs energy. However, previous plasticity methods which do not account for hysteresis only dissipate energy when the elastic limit is exceeded. In our hysteresis based model energy loss is naturally formed by the loop area implying that any bending deformation can dissipate energy, which matches observation of real fabrics where energy is dissipated due to internal friction. In order to show the energy dissipation effect of different methods, two experiments are performed: a piece of hanging cloth is hit by a fast moving sphere (Figure 4.11), and a female character wearing a skirt walks (Figure 4.12). We choose an elastic limit at a low and high curvature, $K_0 = 0.1\pi$ and $K_0 = 0.5\pi$, for the first and second test respectively. Therefore plastic deformation is more easily generated in the test of hitting a cloth. For all simulations, the bending damping is turned off and the air damping coefficient



(a) (b) (c)

Figure 4.12: A human character wearing a skirt is walking. Results of (a) a method without plasticity and hysteresis, (b) plasticity method and (c) our hysteresis model on frame 900. ($b_1 = 3.4, b_2 = 2.0$, c_1 = -0.45, $\lambda = 0.15$ and $\mu = 0.02$)

is very small. The first test shows that the normal method has the strongest oscillations, the plasticity method (similar to [O'Brien et al., 2002]) alleviates this effect, and our model further reduces such behaviour. However, in the second test the plasticity method generates similar results with the normal method in terms of oscillation, because the elastic limit is relatively high so that few bending elements experience plastic deformation. However our method dissipates energy no matter whether plastic deformation occurs or not, leading to a more stable waving behaviour for the skirt. Another difference is that less folds are formed in our method. This is because of not only energy dissipation but also the higher bending stiffness represented by quadratic curves in elastic range. This feature arises intrinsically from the hysteresis model and makes the bending damping component unnecessary. In simulations using explicit integrators, this helps stabilization and thus gives rise to a relative large step size. Static images have difficulty showing these features and thus we refer readers to the accompanying animation.

4.5.4 Fatigue Weakening

In order to show fatigue weakening during plastic deformation, a cyclic test of bending and unbending is designed as follows: after a piece of cloth is first gently folded, then clamped by two parallel planes for a short while, and then the top plane is gradually lifted, followed

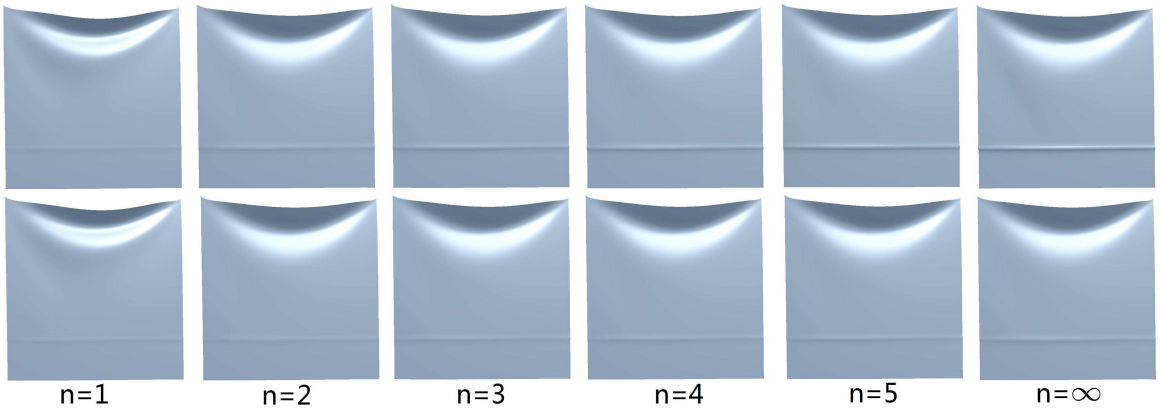
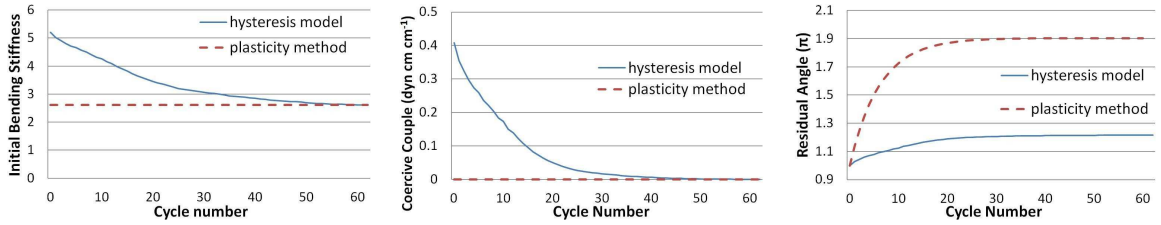


Figure 4.13: A number of pieces of cloth are clamped by two parallel planes n times, and then are hung vertically to show the remained wrinkles, using a plasticity method (top) and our hysteresis model (bottom, $b_1 = 5.20, b_2 = 2.51$*,* c_1 = -4.45*,* $\lambda = 0.15$ *and* $\mu = 0.1$*)*

by another cycle of lowering and lifting the top plane again. The process is repeated for a large number of cycles. A plasticity method without fatigue weakening is also used for comparison. As Figure 4.13 shows, the residual wrinkle increases with cycle number. In our method the first cycle introduces the most residual curvature, and the increase in residual curvature smoothly reduces. Finally, some permanent amount of residual curvature occurs. This observation matches the behaviour discussed in Section 4.3.5. The plasticity method presents a different result since the increase in residual curvatures in each cycle only depends on how much the bending element is bent.

For a single bending element belonging to the wrinkle line, data of the initial bending stiffness, the coercive couple and the residual angle are given in Figure 4.14. The coercive couple is a measure of frictional resistance in textile research. In our system the same amount of interfibre friction occurs no matter the bending direction, thus the coercive couple equals $2|M_0|$. From these curves we can see that the initial bending stiffness starts from the value of b_1 and decreases to b_2, accompanied by the drop in the coercive couple from 0.41 dyn cm/cm to zero. In this experiment, the residual angle increases and finally becomes permanent at around 1.22π. Our hysteresis model includes a natural way of performing fatigue weakening, based on a decreasing difference between the bending and unbending forces, i.e. energy dissipation. Whilst fatigue weakening may be possible with non-hysteresis plasticity models, we are not aware of any work of this kind.



(a) (b) (c)

Figure 4.14: (a) The initial bending stiffness decreases during the cyclic test, asymptoting to b_2; (b) The coercive couple reduces steeply at first, then the rate slowly reaches zero; (c) The increasing residual angle finally remains stable at 1.22π.

	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$
without	1.35ms, 11.4%	2.70ms, 20.5%	4.04ms, 27.9%
with	2.04ms, 16.3%	4.07ms, 28.0%	6.10ms, 36.9%

Table 4.1: Running time and the proportion of the bending component with and without hysteresis, with collision handling turned off, for a piece of cloth consisting of 18,000 triangles. α is the ratio between time steps h_i and h_e.

4.5.5 Performance

In general, collision handling dominates the computation time in cloth simulation, and the bending computation takes a very small amount of total time. When the collision handling is turned off, the simulator spends most of the time executing the PCG solver. The running time and the ratio of the bending component with and without hysteresis for a piece of cloth consisting of 18,000 triangles are shown in Table 4.1. When $\alpha = 1$, our bending model takes 16.3% of the running time. If hysteresis is not considered (hence no plasticity), this number reduces to 11.4%, which means that introducing hysteresis into the system only requires a small increase in the overall computation time. This is important for real time simulation. For example, in our bunny and twist tests (with collision handling turned on) where real time frame rates are achieved, the extra computation time is negligible.

4.6 Conclusion

This chapter describes an efficient physically based bending model using hysteresis in cloth simulation. The bending forces for loading and unloading are treated differently to capture effects of bending plasticity and energy loss due to internal friction, without using any complex friction model at the micro fibre level. Alleviation of oscillations is an intrinsic

feature. Fatigue weakening is investigated based on hysteresis enabling the system to turn increasingly residual wrinkles into permanent ones. Results of various experiments designed to demonstrate the effectiveness of the bending model are presented. Comparisons with previous bending methods and plasticity models are also provided. As cloth simulation lends itself well to parallel architecture, a GPU-based version is implemented to obtain high performance. Performance tests show that only slightly higher computational costs are required for the improved bending model.

Chapter 5

Virtual Subdivision for GPU-based Collision Detection

The main challenge that arises for collision detection of highly deformable objects, such as cloth, is the large number of primitives that may collide with each other in any given time step. Although a number of approaches have been investigated, performance is still a major bottleneck. GPU computing has demonstrated its potential benefit to perform physics calculations, including collision detection. Recent work has been focusing on designing fast algorithms based on bounding volume hierarchies (BVHs) and spatial subdivision schemes. BVHs have long been used to accelerate collision detection in simulation of deformable objects. However, their tree-based hierarchical data structure is a challenge to efficient parallelism. In contrast, spatial subdivision methods exhibit very high data parallelism, well suited to the GPU's parallel architecture. Generally spatial subdivision methods suffer from two main difficulties: uneven triangle sizes and uneven triangle distributions, both of which can easily impair subdivision efficiency, leading to a large number of broad phase tests. This chapter addresses the first problem using a virtual subdivision scheme to present an improved uniform grid. The second issue is discussed in the next chapter.

5.1 Introduction

Spatial subdivision has been widely used to accelerate collision detection (CD) by significantly reducing the number of collision tests. The main strategy is to partition the space into many cubic cells, and then perform collision tests only for primitive pairs belonging to the same cell, resulting in an average complexity of $O(n)$ in a scene containing n features such as

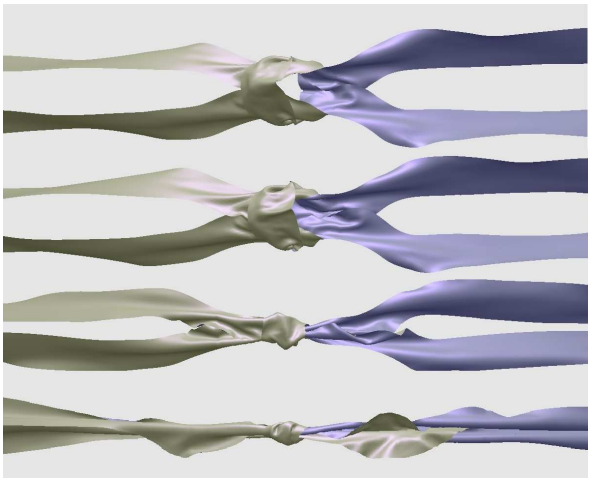


Figure 5.1: Snapshots from the reef knot benchmark: Two pieces of ribbon are tied into a reef knot.

triangles and their bounding volumes (BVs). Early spatial subdivision based CD approaches focus on simulation of rigid objects, such as [Zhang and Yuen, 2000], [Mirtich, 1997] and [Bandi et al., 1995]. Later, a number of methods employing spatial subdivision for CD of deformable objects including cloth have been proposed. In general, these approaches use a uniform grid whose cell size is object-independent [Teschner et al., 2003], a uniform grid with an object-dependent cell size [Pabst et al., 2010], a two-level hierarchical grid [Fan et al., 2011] or a general multi-level hierarchical grid [Eitz and Lixu, 2007]. Recently, several papers have focused on utilizing the GPU's parallelism to achieve high performance for CD by taking advantage of the features of spatial subdivision [Pabst et al., 2010], [Liu et al., 2010] and [Fan et al., 2011].

The main drawback of methods employing a traditional uniform grid is that an inappropriate cell size can lead to a very high number of false positive CD tests. Triangles may overlap too many cells if the cell size is small, or too many triangles can belong to the same cell when the cell size is large. Moreover, an appropriate cell size may not exist when triangle sizes are uneven. Unfortunately, it is unrealistic to expect a mesh with evenly sized triangles in many scenarios. One way to alleviate this limitation is the hierarchical grid of multiple cell levels where each primitive is assigned to a level whose cell size fits well [Eitz and Lixu, 2007; Kroiss, 2013], but this technique does not adjust the hierarchy according triangle sizes and has to perform additional computation for CD across different levels. In addition, the simplicity and scalability of the uniform grid are more suited to the nature of

the GPU architecture.

As discussed in Chapter 2, another issue is the removal of duplicate CD tests due to object pairs assigned to multiple cells. There are a number of existing methods to address this issue, such as running iteration steps, checking objects against others appearing in adjacent cells, and utilising the location of the minimal vertex of the overlapped axis-aligned bounding box (AABB). Without relatively expensive operations, Grand [2007] cheaply handles this problem in parallel on GPUs using control bits (CBs). In this method, a strategy that sets the cell size greater than the largest feature ensures that a feature can be assigned to up to eight cells in 3D space, so one integer per object is enough to specify the cells overlapped by this object (one bit for one cell). In the subsequent broad phase CD, all that is needed is to perform cheap bit operations. However, because it only works for grids with an object-dependent cell size adapted to the largest object, the problem that cells can contain too many triangles still exists if the size of largest feature is much greater than the average feature size.

In order to avoid the relative large cell size and still take the advantage of CBs, this chapter presents a virtual subdivision scheme (VSS) for uniform grid based CD algorithm to use a more optimal cell size by subdividing large triangles into small child triangles every frame. The subdivision is virtual because virtual child triangles are only used for the purpose of CD in the broad phase, and the newly generated vertices and edges are not used in the simulation. Therefore, reconstruction of the triangle mesh and the linear system for the simulation is not required. We do not retain the refined triangle mesh so that the input mesh is virtually subdivided on the fly each frame. In our GPU-based implementation, the virtual subdivision can be quickly completed. Most duplicate collision tests in the broad phase caused by triangle subdivision are efficiently avoided. Although some child triangles are generated in the VSS approach, we demonstrate that this method dramatically reduces the number of broad phase collision tests and the GPU memory requirement due to the smaller cell size, and thus increases the overall performance. This method can work with both discrete and continuous collision detection (DCD and CCD). Results show that the method provides speedups by comparing performance with existing methods.

The remainder of the chapter is organised as follows. We start by briefly reviewing previous work directly related to spatial subdivision based CD methods in Section 5.2. The virtual subdivision approach is presented in Section 5.3, followed by discussion of results and performance in Sections 5.4. Finally, the conclusion and future work are given in Section 5.5.

5.2 Related Work

In this section, previous work directly related to spatial subdivision based CD methods are briefly introduced. For more detail information about CD, please refer to Chapter 2 and 3.

Teschner et al. [2003] present a technique using a uniform grid with spatial hashing for CD of deformable objects consisting of tetrahedral meshes. In this method, the cell size is object independently chosen. They discussed how a set of parameters and properties affect the performance, and their results suggest that the best performance is achieved when the cell size is set to about the same as the average size of the AABBs. Eitz and Lixu [2007] extend it to a method with a hierarchical grid. Grand [2007] employs a uniform grid with an object dependent cell size to perform CD on GPUs, in where the CBs scheme is introduced. Recently, based on the above technique, Pabst et al. [2010] present a hybrid CPU/GPU approach focused on deformable triangle meshes. This chapter addresses the primary shortcoming identified in their work. Kroiss [2013] partially solves this issue using a hierarchical grid with multiple cell levels on GPUs. Other spatial subdivision techniques for CD are discussed in [Alcantara et al., 2009] and [Fan et al., 2011].

5.3 Approach

In this section a virtual subdivision based CD approach is presented. In the broad phase of our CD pipeline, BV tests are performed to cheaply select triangle pairs that need to undergo further intersection tests in the narrow phase. In the narrow phase, proximity tests used in [Bridson et al., 2002] are performed to compute the collision and intersection information of vertex-triangle (VT) and edge-edge (EE) pairs. For DCD, tests are carried out to calculate intersections at each discrete time step. In the case of CCD, the path of moving primitives between two time steps is considered to find out the first contact instant by testing coplanarity, which reduces to solving a cubic equation [Provot, 1997], with proximity tests. Therefore, the BV for CCD has to bound the trajectory of the moving primitive during the time interval. We use two types of BV. Spheres are used to determine triangle assignment and also to identify triangles which need to be subdivided. K discrete oriented polytopes (K-DOPs [Klosowski et al., 1998], specifically 18-DOPs) are used for BV tests in the broad phase. Algorithm 1 shows the overview of the VSS method. Italic lines relate to the core contribution of this chapter.

Algorithm 1 Collision Detection

update BVs of triangles, edges and vertices
update cell size of the adaptive uniform grid
while *triangle list is not empty* **do**
  for all *triangles* **do**
    determine if subdivision is required
    subdivide it if necessary and put child triangles into the list
    remove it from the list
  end for
end while
for all triangles that are not subdivided **do**
  assign to cells overlapped by it
end for
sort cell-triangle pairs
remove redundant cell-triangle pairs
for all triangle pairs in cells **do**
  CBs test to remove duplicate collision tests
  triangle-based BV culling
  R-triangles culling
  feature-based BV culling
  output candidate feature pairs (VT/EE)
end for
for all output candidate VT and EE pairs **do**
  proximity test
end for

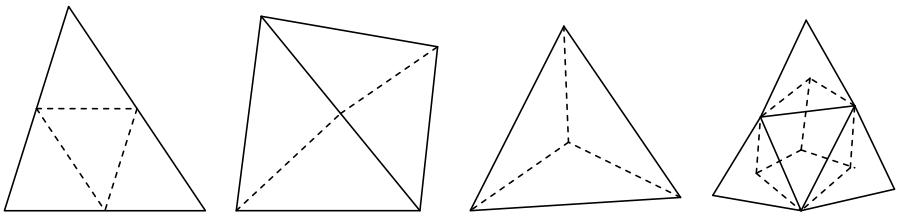
5.3.1 Cell Size

Instead of using a cell size adapted to the largest triangle, we define the cell size according to the average triangle bounding sphere size:

$$size = 3 \times \lambda r_a \tag{5.1}$$

where r_a is the average radius of triangles' bounding spheres in the current frame. λ is a coefficient which must be greater than one in order to avoid subdividing too many triangles. We empirically set it to a value between 1.1 and 1.3. Following [Grand, 2007], we scale λr_a by 3 to enable the use of CBs. Now, with the AABB of the entire model, three dimensions along the X, Y and Z axes can be easily computed.

If λ is set too large or small, the improvement of the VSS method in the performance can



(a) Loop scheme (b) Tri-quad (c) $\sqrt{3}$-subdivision (d) $\sqrt{3}$-refinement

Figure 5.2: Some well known triangle subdivision rules

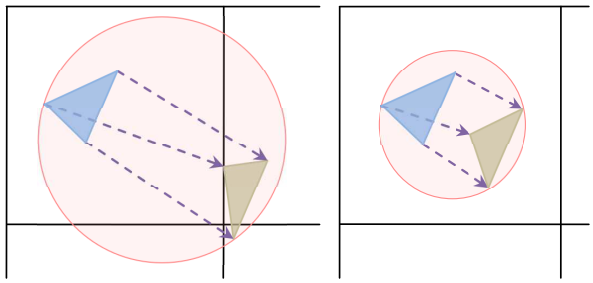


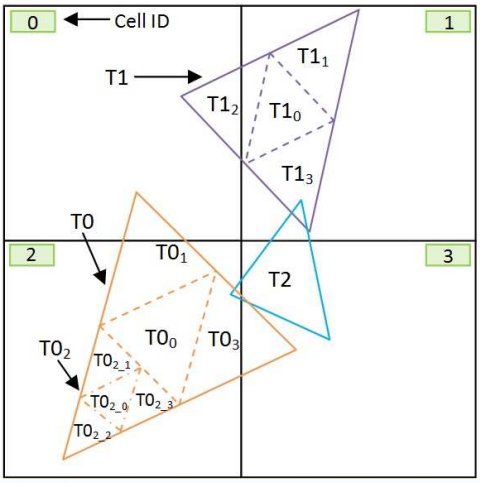
Figure 5.3: In this case (left), the large time step needs to be halved to shorten the trajectory of the triangle (right). Otherwise, the BV is always too big.

degrade. However, we found this simple scheme works well in our benchmarks, as it is not very performance sensitive if λ is in the range from 1.1 to 1.3, and we use 1.2.

5.3.2 Triangle Subdivision

Since this method is designed to allow the BV of each triangle to overlap no more than eight cells, more refined triangles are needed for large triangles in every time step. Some well known subdivision rules have been proposed in level-of-detail and multi-resolution areas, including the Loop scheme [Loop, 1987], tri-quad subdivision [Velho and Zorin, 2001], $\sqrt{3}$ subdivision [Kobbelt, 2000] and $\sqrt{3}$-refinement [Alliez et al., 2003] (Figure 5.2). Since we are interested in evenly refining triangles, the Loop scheme is the best choice, as it subdivides a triangle into four equal child triangles by taking midpoints of the original edges (Figure 5.2(a)).

In each frame, once the grid cell size and BVs are updated, we first decide if each triangle has to be subdivided by checking whether the radius of its BV is greater than one third of the cell size, calculated as discussed above, and then split those marked as needed. For newly created child triangles, their BVs are computed, and their original root triangles are stored as well. This subdivision operation is repeated until there are no triangles needing to be subdivided. In our benchmarks, the number of subdivision levels needed is up to four.



	T0	T1	T2	$T0_0$	$T0_1$	$T0_2$	$T0_3$	$T1_0$	$T1_1$	$T1_2$	$T1_3$	$T0_{2\_0}$	$T0_{2\_1}$	$T0_{2\_2}$	$T0_{2\_3}$
Tri. ID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Subdivided	O	O	X	X	X	O	X	X	X	X	X	X	X	X	X
Sub. level	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2

Figure 5.4: After the virtual subdivision is done, a new triangle ID list is created. In each subdivision level, child triangles from the same input triangle are coherent. Triangles marked as red cross are assigned to cells.

However, when it comes to CCD, the time step needs to be taken into account. Because collisions are checked for moving objects between two time steps in CCD, the BV covers the triangle's trajectory from the beginning to the end of the current time step. Hence, the condition to quit the subdivision step will never be satisfied, if the time step is not small enough (see Figure 5.3). We solve this problem by repeatedly halving the time step to shorten the trajectory of movement and thus decreasing the swept volume of BVs.

Since the cell size is updated at the beginning of every frame and triangles undergo deformation during the simulation, we dynamically perform the virtual subdivision every frame and do not keep the refined virtual mesh after CD is done. Due to the independency of the Loop scheme, a CUDA kernel is invoked to simultaneously perform the subdivision with each thread handling exactly one triangle. Once this step is done, a new triangle ID list is generated (see Figure 5.4). Then, triangles (including input and newly created child triangles) that do not need to be further subdivided are assigned to cells (a cell-triangle array is built), and their CBs are computed. Note that, child triangles are used to represent their root triangles in the broad phase only, and in the narrow phase we still check proximity for VT and EE pairs using original input triangles. Therefore, the physics simulation and collision response are still computed using the original mesh to avoid expensive computations

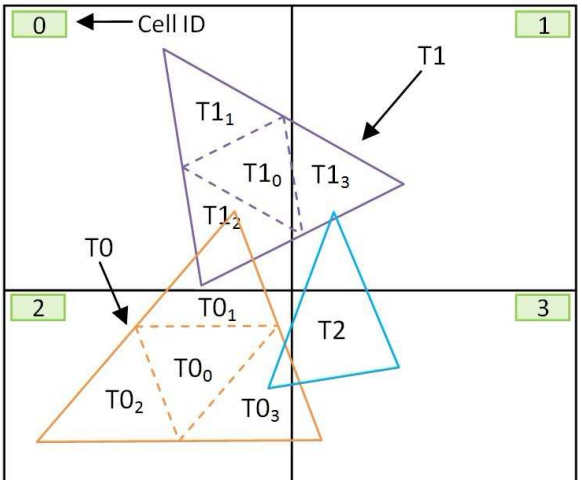


Figure 5.5: **First issue:** *In cell*0*, collision test between* $T0$ *and* $T1$ *will be performed four times if all child triangles of* $T1$ *are assigned to cell*0*.* **Second issue:** *Since the centroid of* $T1_3$ *is in cell*1*, it has to be the representative of* $T1$ *to the cell, so all triangles overlapped by cell*1 *will test against* $T1$ *in the broad phase. Otherwise, the test between* $T1$ *and* $T2$ *is ignored.*

due to the refined mesh.

5.3.3 Remove Redundant Cell-triangle Registration

Duplicate collision tests might be introduced, when child triangles belonging to a root triangle are assigned to the same cell. For example (see Figure 5.5), four tests referring to the same root triangle pair ($T0$, $T1$) are produced in *cell*0 because the four child triangles of $T1$ are assigned to this cell. Clearly, a collision test for the triangle pair ($T0$, $T1$) needs to be executed only once, which can be achieved by registering only one child triangle to a cell for each root triangle. However, there is one thing to be noticed: the child triangle with its centroid located in a cell should be the one given priority, otherwise collisions might be missed because, according to the approach in [Grand, 2007], triangles assigned to the same cell but having their centroids located in surrounding cells are not tested against each other in that cell. For example, in Figure 5.5, if $T1_0$ is assigned to *cell*1, the collision between $T1$ and $T2$ will be missed due to the fact that neither of them ($T1_0$, $T2$) have their centroids belonging to this cell. Therefore, $T1_3$ must be assigned to *cell*1 to represent its root triangle $T1$.

To eliminate these duplicate tests caused by triangle subdivision, several parallel radix sort [Harris, 2007] runs are performed on the cell-triangle array. Then a kernel is invoked

Triangle	$T0_0$	$T0_1$	$T0_1$	$T0_2$	$T0_3$	$T0_3$	$T1_0$	$T1_0$	$T1_1$	$T1_2$	$T1_3$	$T1_3$	T2	T2	T2
Cell ID	2	0	2	2	2	3	0	1	0	0	0	1	1	2	3
Home/Phantom	H	H	P	H	H	P	H	P	H	H	P	H	P	P	H

Sort

Triangle	$T0_1$	$T1_0$	$T1_1$	$T1_2$	$T1_3$	T2	$T1_3$	$T1_0$	T2	$T0_0$	$T0_2$	$T0_3$	$T0_1$	T2	$T0_3$
Cell ID	0	0	0	0	0	1	1	1	2	2	2	2	2	3	3
Home/Phantom	H	H	H	H	P	P	H	P	P	H	H	H	P	H	P
Valid	O	O	X	X	X	O	O	X	O	O	X	X	X	O	O

Figure 5.6: This figure shows the cell-triangle array for Figure 5.5, where Home indicates that the centroid of the triangle locates inside the associated cell, and Phantom indicates that the centroid of the triangle does not locate inside the associated cell. Parallel radix sorts are performed to find out items that have the same cell ID and refer to the same root triangle. The duplicate registrations (marked as the red cross) are removed.

Triangle	$T0_1$	$T1_2$	T2	$T1_0$	$T1_1$	$T1_3$	$T1_2$	T2	$T0_0$	$T0_1$	$T0_3$	$T0_{2\_0}$	$T0_{2\_1}$	$T0_{2\_2}$	$T0_{2\_3}$	T2	$T0_3$
Cell ID	0	0	1	1	1	1	1	2	2	2	2	2	2	2	2	3	3
Home/Phantom	P	H	P	H	H	H	P	P	H	H	H	H	H	H	H	H	P
Valid	O	O	O	O	X	X	X	O	O	X	X	O	X	X	X	O	O

Figure 5.7: This figure shows the cell-triangle array after one parallel radix sort is performed for Figure 5.4. Most duplicate, redundant cell-triangle registrations that have the same cell ID and root triangle are marked as red cross to remove. Among remains, only registrations of $T0_0$ and $T0_{2\_0}$ are duplicated.

to quickly scan the sorted array and remove duplicate items having identical cell ID and root triangle ID (see Figure 5.6). This parallel radix sort dependent scheme can remove all redundant cell-triangle pairs, but is a little expensive due to the multiple sort runs. Alternatively, we can take advantage of the fact that in each subdivision level child triangles belonging to the same root triangle are coherent in the triangle list (see Figure 5.4). Hence, after only one radix sort operation by the cell ID, most duplicate cell-triangle pairs can be found and removed (see Figure 5.7). In fact, this sort operation is originally needed for a traditional uniform grid based method even without the virtual subdivision scheme to put items with the same cell ID together [Grand, 2007], which means our removal step only requires launching a kernel to scan the sorted array to find and delete duplicate registrations. In practice, this scheme is simpler and faster but also can avoid most of the redundant collision tests caused by duplicate cell-triangle pairs.

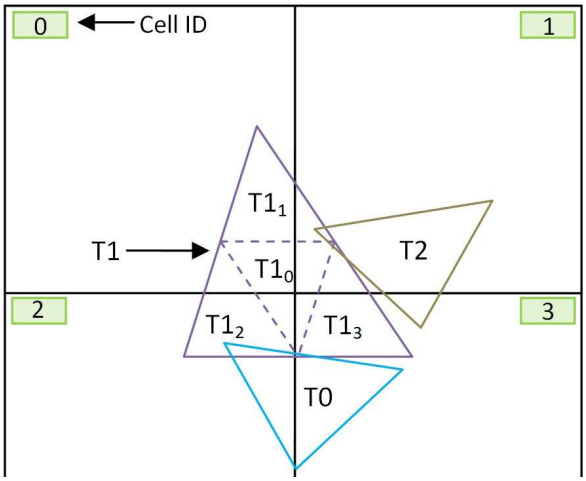


*Figure 5.8: In cell*1*, cell*2 *and cell*3*,* $T1_0$*,* $T1_2$ *and* $T1_3$ *are chosen to be assigned to represent the root triangle* $T1$ *respectively, and we assume that their centroids are in cell*0*, cell*2 *and cell*3 *respectively.*

However, another mechanism is needed to remove duplicate tests introduced by the situation illustrated in Figure 5.8. In this example, because triangle $T1_2$ and $T1_3$ are assigned to *cell*2 and *cell*3 respectively, the test referring to $T0$ and $T1$ is redundantly performed twice in two cells. The same problem happens to triangle $T1$ and $T2$. To solve this problem, we expand the CBs of child triangles to contain the cell type (different from cell ID, refer to Chapter 3 for the definition) of the centroid of sibling triangles. Then, by checking CBs of the two involved triangles we keep the test associated with the child triangle both having its centroid located in the collision cell and the smallest child ID, and remove others. Therefore, in the case of the triangle pair ($T0$, $T1$), the test that occurs in *cell*2 is executed and the test in *cell*3 is abandoned. For the triangle pair ($T1$, $T2$), the test belonging to *cell*3 is retained and the test in *cell*1 is removed because $T1_3$ is the only child triangle with its centroid located in the collision cell (*cell*0 contains $T1_0$'s centroid). Notice that similar duplicate tests might be introduced by large triangles subdivided more than once (three or more subdivision levels), and can not be handled by the proposed method. However, these duplicate tests rarely occur in practice, and in our benchmarks the number of such tests is less then 0.4% of the total broad phase collision tests.

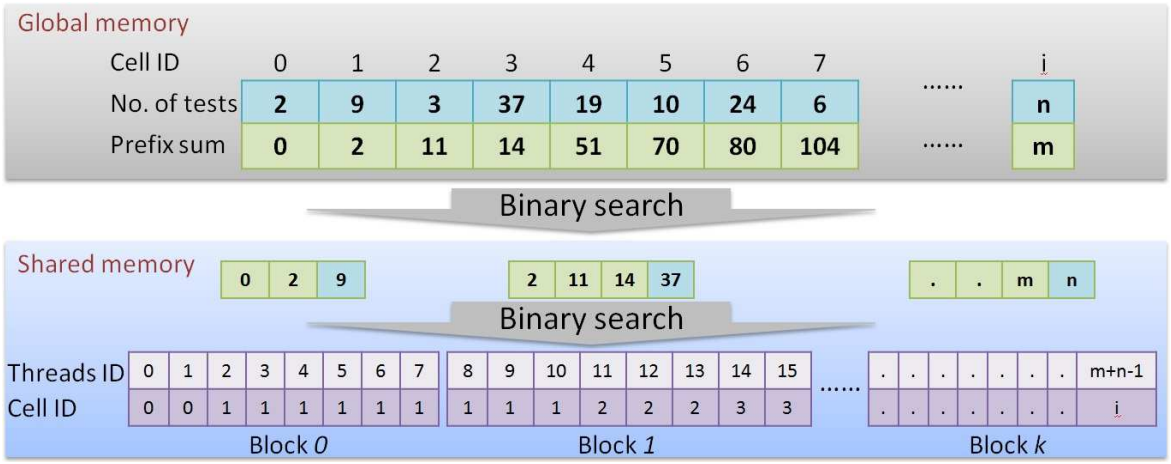


Figure 5.9: The diagram of the workload distribution scheme.

5.3.4 Workload Distribution

The number of collision tests for each cell is determined according to the assigned triangles [Pabst et al., 2010]. In practice, cells are very unlikely to have a similar number of collisions. Therefore a GPU implementation will give poor performance if a single thread is launched to handle all collisions occurring in a cell, as threads with less work wait for threads having more work. In order to obtain the best performance in a parallel system, it is necessary to balance computation load among concurrent threads. We achieve this by assigning one unique collision test to a thread. However, a mechanism is needed to determine which cell the collision test handled by each thread belongs to. Pabst et al. [2010] proposed a scheme based on constructing and traversing a tree of degree 16 which contains the number of collision tests per cell. Fan et al. [2011] addressed this issue without maintaining such a complex data structure by applying a parallel scan [Sengupta et al., 2007] on the array of the number of tests per cell and another segmented parallel scan on the workload array. However, the length of the workload array can be up to several tens of millions in a simulation consisting of large models, and thus a large amount of GPU memory is needed for the parallel scan. Input, output and flag arrays of length ten million require 114MB memory on the GPU. In experiments with complex objects, this scheme could easily make current graphics cards hit the physical memory limit.

We present a new scheme to evenly distribute workload (see Figure 5.9)), which does not need any additional memory. Although this workload distribution method is approximately

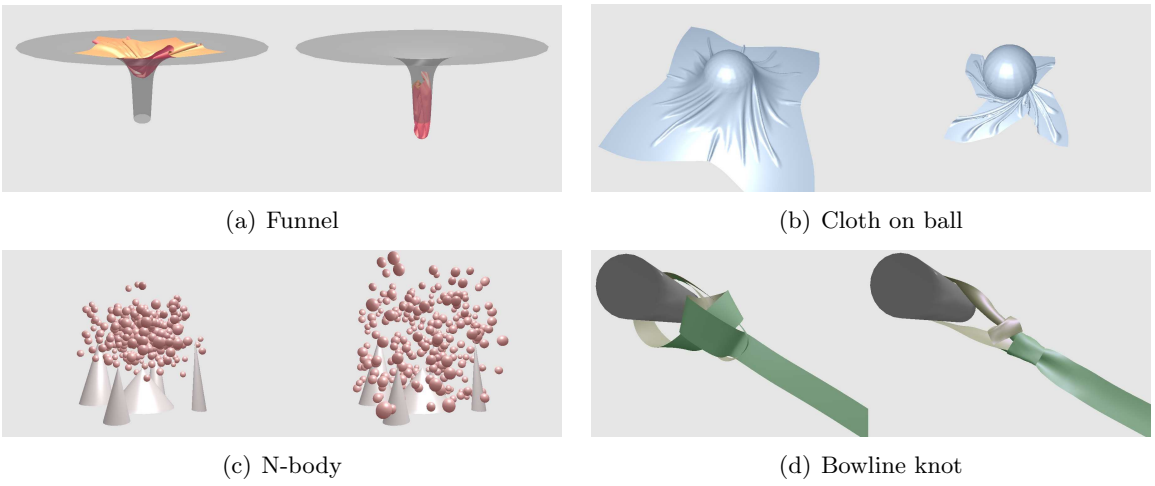
40% slower than the scheme in [Fan et al., 2011], the impact of this difference on the overall performance is trivial as long as each thread is allocated to a single test, because the balancing process itself takes a very small part of the total running time. This can be an important trade-off for systems with a huge amount of data.

The first step is to perform a parallel scan on the array that contains the number of collision tests of each populated cell to generate a prefix sum array just as in [Fan et al., 2011]. Now for each populated cell, the total number of tests occurring in all previous cells is known. We then invoke as many threads as the total number of collision tests, each thread having an unique ID corresponding to a unique test. For the first and last thread of each thread block, a binary search is used with its thread ID to find the position/cell ID (where the collision test occurs) in the prefix sum array in global memory. Because of the continuity, cell IDs corresponding to remaining threads in each block are between cells of the first and last thread, which means values of this sub prefix sum array can be copied into shared memory, and all remaining threads use binary search to quickly find which cell their tests belong to.

5.3.5 Collision Test

After the cell index is determined, the triangle pair involved in the collision test can be easily found. Before performing the broad phase CD, cell data needed for the test is loaded into shared memory for fast access. In the broad phase test, for a triangle pair we first use their CBs to check if this test is redundant, so the CD for triangle pairs appearing in multiple cells is guaranteed to be processed only once. The next step is the BV test, for which BVs of original triangles are used, because not all child triangles are assigned to cells. Otherwise collisions would be missed if the BV of child triangles are used. Once this step is completed use of virtual subdivision is finished.

In the subsequent steps, R-triangle tests proposed in [Curtis et al., 2008] are first performed to remove duplicate VT and EE elementary tests introduced by the fact that a vertex or an edge can be shared by more than one triangle in a mesh. Before outputting VT and EE pairs for more expensive elementary tests in the narrow phase, the vertex and edge based BV culling is carried out to further eliminate false positives. In the narrow phase, for both DCD and CCD, proximity tests [Bridson et al., 2002] to compute exact intersection between VT and EE pairs are carried separately by invoking two different kernels to avoid branching and divergence. For CCD, coplanarity is checked by solving a cubic equation before the proximity test.



(a) Funnel (b) Cloth on ball

(c) N-body (d) Bowline knot

Figure 5.10: This figure shows the snapshots from benchmarks used for performance measuring. All benchmarks are tested on a computer with an NVIDIA GTX 470 GPU.

5.4 Results and Performance

We implemented the method using the CUDA toolkit 4.1 and conducted experiments on a computer with an NVIDIA GTX 470 card. All CD steps are performed on the GPU without data transfer between CPU and GPU during the simulation, and so the CPU and main memory do not affect performance.

5.4.1 Benchmarks

In order to investigate the performance of the VSS method, we used a set of benchmarks with different scenarios. These benchmarks consist of triangles whose sizes vary considerably. Refer to Online Resource 1 for the animation.

1. Funnel (18K triangles): A piece of cloth falls into a narrow funnel and goes through it, causing many self collisions. (Figure 5.10(a))

2. Cloth on ball (92K triangles): A cloth with a large number of triangles drapes on a rotating ball. (Figure 5.10(b))

3. Flamenco (49K triangles): A dancing model wearing a multi-layer skirt. (Figure 5.11)

4. N-body (146K triangles): Hundreds of balls are colliding with each other and cones. (Figure 5.10(c))

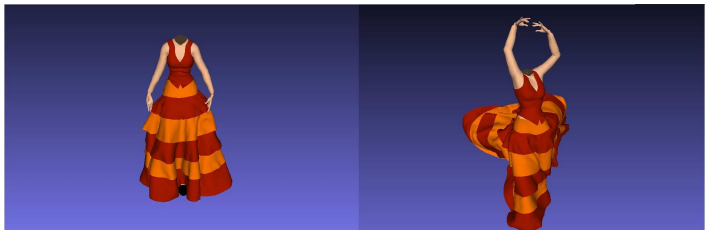


Figure 5.11: Snapshots from the flamenco benchmark.

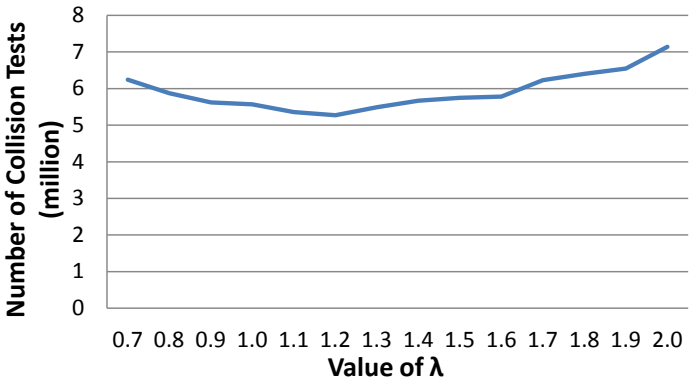


Figure 5.12: The number of tests performed in the broad phase for the cloth on ball benchmark with varying λ.

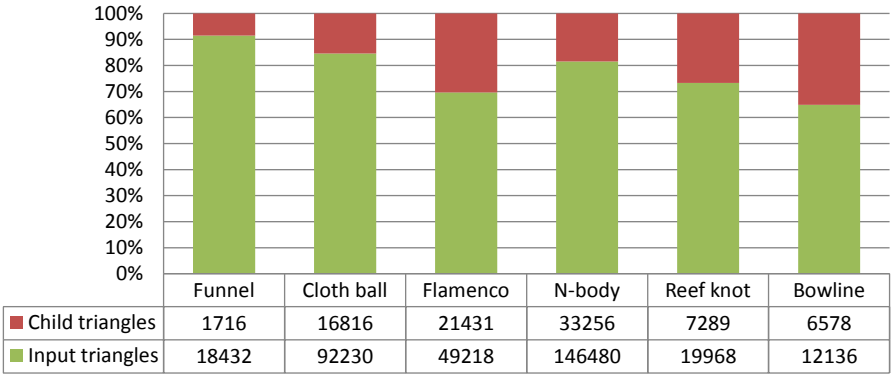
5. Reef knot (20K triangles): Two pieces of ribbon are tied into a reef knot. (Figure 5.1)

6. Bowline knot (11K triangles): One piece of ribbon is tied into a bowline knot. (Figure 5.10(d))

Motion data of the *cloth on ball*, *flamenco* and *N-body* tests were provided by the UNC Dynamic Scene Benchmarks collection. For other tests, the spring-mass method is used to compute cloth internal dynamics with the implicit backward Euler scheme (Equations 3.25, 3.26 and 3.27) for simplicity. Since the work does not focus on the modelling of cloth internal forces, the mass-spring method satisfies the needs of our work.

5.4.2 Choice of λ

The VSS method avoids using a cell size adapted to the largest object, but as mentioned in Section 5.3.1, the value of λ influences the number of subdivided triangles and triangles



	Funnel	Cloth ball	Flamenco	N-body	Reef knot	Bowline
Child triangles	1716	16816	21431	33256	7289	6578
Input triangles	18432	92230	49218	146480	19968	12136

Figure 5.13: The figure shows the number of newly created child triangles and the ratio between the number of child triangles and total triangles for CCD.

assigned to each cell, which determine how many collision tests are processed in the broad phase, and thus the overall performance. The number of collision tests performed in the broad phase for the *cloth on ball* benchmark with varying λ is given in the Figure 5.12. The measurement indicates that the number of tests increases if λ is too small or too large, and the minimum number of tests is obtained when λ is set to near 1.2. A similar situation is observed for other benchmarks. Therefore, for the following results a value of 1.2 is used.

5.4.3 Performance and Analysis

The primary goal of the VSS method is to efficiently reduce the false positives resulting from a large cell size to improve overall performance. The average number of child triangles created per frame is given in Figure 5.13, and Figure 5.14 compares the average number of invoked broad phase collision tests per frame between the VSS method and a traditional uniform grid method without the virtual subdivision scheme. For all benchmarks the ratio between the number of newly generated child triangles and total triangles is fairly small, from 9% to 35%, but yields approximately an order of magnitude reduction of broad phase collision tests for several of our test cases. In the *N-body* benchmark, for example, the traditional method produces around 12 times as many broad phase tests as the VSS method. This is because cones consist of triangles which are significantly larger than the triangles in spheres, so that a large cell size is used in the traditional method and a high number of collision tests have to be processed, while the VSS method is intended to avoid this issue. A similar situation arises with other benchmark models.

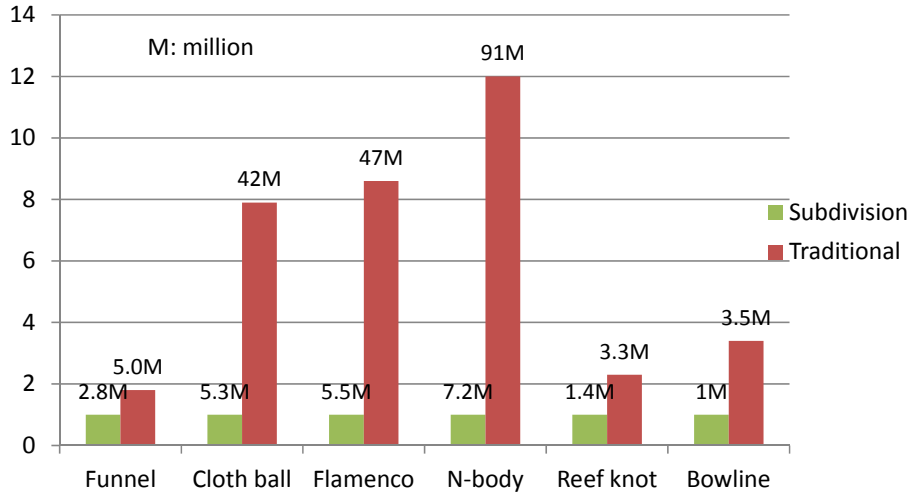


Figure 5.14: Comparison of the average number of broad phase tests between the VSS method and a traditional uniform grid method for CCD. A significant reduction is observed, and thus less memory is required.

Another aspect of reducing the number of collision tests in the broad phase is the ability to run larger scenes. In a scenario that contains complex models undergoing large scale deformation, memory usage becomes one of main issues on GPUs. For instance, if the virtual subdivision is taken out from the VSS method, the *N-body* experiment will run out of GPU memory on our machine as an array of length over one hundred million is needed to store triangle pairs in some frames. In the VSS method with virtual subdivision, an array size less than ten million is adequate.

Running times of each step for the *N-body* benchmark are presented in Figure 5.15. The virtual subdivision step takes just 8% of the total running time, and the method spends most of the time executing broad phase tests (48%). The performance of the VSS approach (including all CD operations) for DCD and CCD on benchmarks is given in Table 5.1.

5.4.4 Comparison

We first compare the performance of the VSS approach against two recent grid based CD algorithms, followed by a comparison with previous bounding volume hierarchy (BVH) based methods. A complication arises from different hardware. The specifications of graphics cards and GPUs discussed are given in Table 5.2.

Pabst et al. presented a CPU/GPU hybrid method extended in [Grand, 2007] to perform

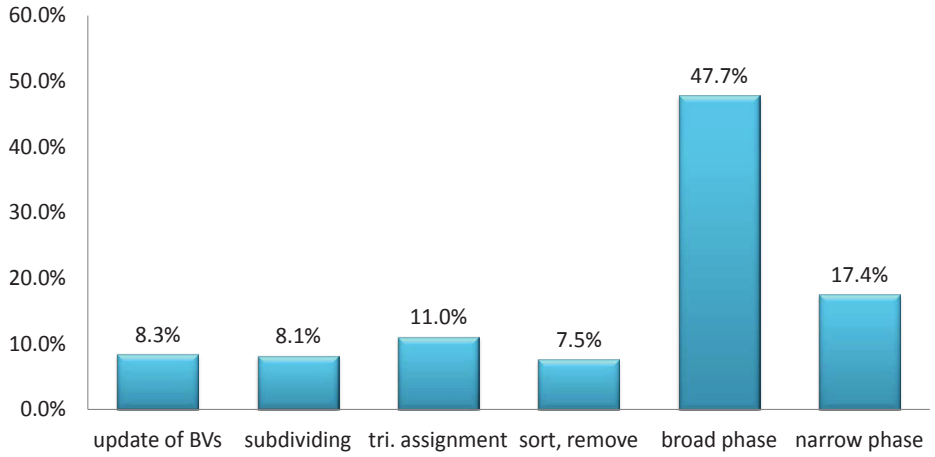


Figure 5.15: Time ratio of steps for N-body benchmark.

	No. of Triangles (×1000)	DCD (ms)	CCD (ms)
Funnel	18	5.1	5.6
Cloth on ball	92	12.0	14.8
Flamenco	49	14.2	16.1
N-body	146	26.7	31.2
Reef knot	20	5.5	6.0
Bowline knot	12	4.3	4.0

Table 5.1: The average timings of the VSS method for both DCD and CCD.

CD on deformable objects consisting of triangle mesh. As discussed by Pabst et al., the cell size issue is one of the main limitations of their method. Since the VSS method is designed to address this primary shortcoming, a comparison is quite straightforward, except for hardware differences, and given in Figure 5.16. They measured the performance using 1, 2 and 4 GPUs on two GTX 295 cards, each of which has two GPUs. The performance of the VSS method for both DCD and CCD on all benchmarks better than their results using 1 and 2 GPUs (speedups of 5.9X and 3.1X are yielded for the *N-body* and *cloth on ball* test). Compared to their best performance measured on four GPUs, the VSS method using one GPU is faster in all except for the *funnel* benchmark, because we used a higher resolution mesh containing more triangles (18K vs. 7.6K). An improvement of a factor of 2.5X is achieved for CCD of *N-body* test.

Fan et al. [Fan et al., 2011] presented a two level hierarchical grid based method to

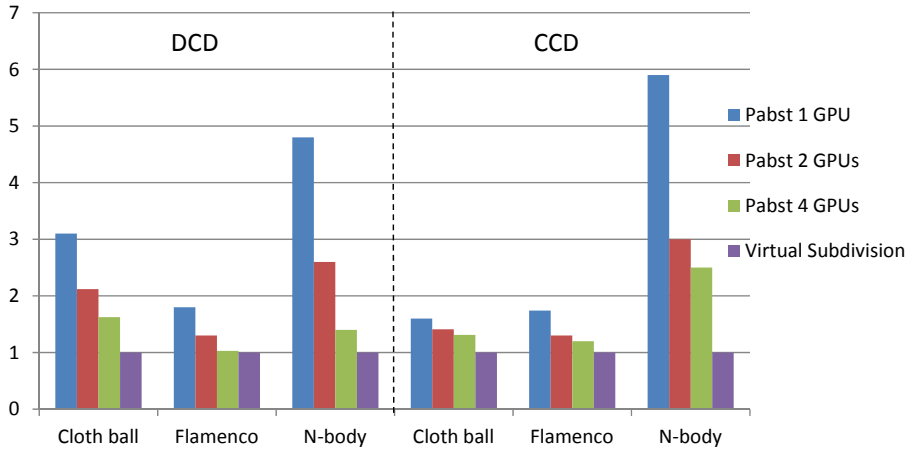


Figure 5.16: The performance comparison between the VSS approach using a GTX 470 GPU and the method proposed by Pabst et al. [Pabst et al., 2010] using 1, 2 and 4 GTX 295 GPUs.

	GTX295	GTX470	GTX480
CUDA Cores	2*240=480	448	480
Graphics Clock(MHz)	576	607	700
Processor Clock(MHz)	1242	1215	1401
Memory Clock(MHz)	999	1674	1848

Table 5.2: Specifications of three different graphics cards.

subdivide the space. This method reduces the false positives by performing a second spatial subdivision operation on dense cells, to split them into eight sub-cells. However, the step to build the hierarchy is quite expensive as it takes around 20% running time. Although they run experiments on a faster GPU (GTX 480), the VSS method is faster for CCD on *funnel*, *cloth on ball* and *N-body* tests (the improvement is up to a factor of 1.5X). Since proximity tests in the narrow phase of DCD are performed to check collisions between VT and EE pairs in the VSS method and they do collision tests only on triangle pairs for DCD, a comparison of DCD timings is not provided.

Among algorithms using BVHs to accelerate CD we are aware of, the fastest multi-core CPU-based and GPU-based methods are presented in [Tang et al., 2009] and [Tang et al., 2011] respectively. Only CCD timings are provided for these two methods. The fastest CPU-based method [Tang et al., 2009] reports timings for *flamenco* and *cloth on ball* benchmarks on a Intel Xeon PC with four quad-core CPUs (so 16 cores) at 2.93 GHz of 27ms and 32.5ms, which are significantly slower than ours at 16.1ms and 14.8ms. Compared to the performance

of the fastest GPU-based approach [Tang et al., 2011] where benchmarks were run on a GTX 480 GPU, with a slower GPU the VSS method works around 2.0X and 2.5X times faster on *flamenco* and *N-body* respectively (16.1s vs. 32.7ms, and 31.2ms vs. 79ms).

5.5 Conclusion

We presented an extended uniform grid based approach on GPU for DCD and CCD of deformable object simulation, which uses a virtual subdivision of the triangle mesh. This method efficiently alleviates the issue of uneven triangle sizes, one of the common limitations of spatial subdivision techniques, by virtually subdividing large triangles to generate a mesh with relatively even triangle sizes, enabling the use of a smaller, better suited cell size. In the most expensive stage, expanded CBs and a transition scan are used to quickly eliminate most duplicate collision tests in the broad phase. In terms of the nature of GPU features, the VSS method provides improvements in the programmability and extensibility, and also reduces the memory usage. In addition, the core idea in this chapter can be combined with other spatial subdivision methods with an object-dependent cell size. Performance improvements are observed by comparing results with those of prior methods.

Chapter 6

An Adaptive Octree Grid for GPU-based Collision Detection

The previous chapter describes a GPU-based collision detection method that uses a virtual subdivision scheme with a uniform grid to address the issue of uneven triangle sizes, which is one of common difficulties in spatial subdivision based collision detection methods. However the other issue, uneven triangle distributions, still presents difficulties for methods using uniform grids. This chapter presents a GPU-based approach to efficiently address this problem by adaptively partitioning space according to triangle distribution using an octree grid.

6.1 Introduction

As discussed in the previous chapter, spatial subdivision based collision detection (CD) methods usually suffer from two main difficulties: uneven triangle sizes and uneven triangle spatial distributions, both of which can easily impair the subdivision efficiency, leading to a large number of broad phase tests. This chapter focuses on the latter one. Generally, uniform subdivision works well for even triangle distributions, but it can hardly partition space with an appropriate cell size when spatial distributions are uneven, because a uniform cell size can be too small for low density areas, and too large for high density areas, resulting in an inefficient subdivision (Figure 6.1). In contrast, non-uniform subdivision techniques can alleviate limitations resulting from uneven spatial distributions. These subdivision techniques mainly include hierarchical grids and octrees.

Hierarchical grids have been adopted for previous GPU-based approaches, due to the suitability for GPUs. Some authors have presented approaches using this technique to address

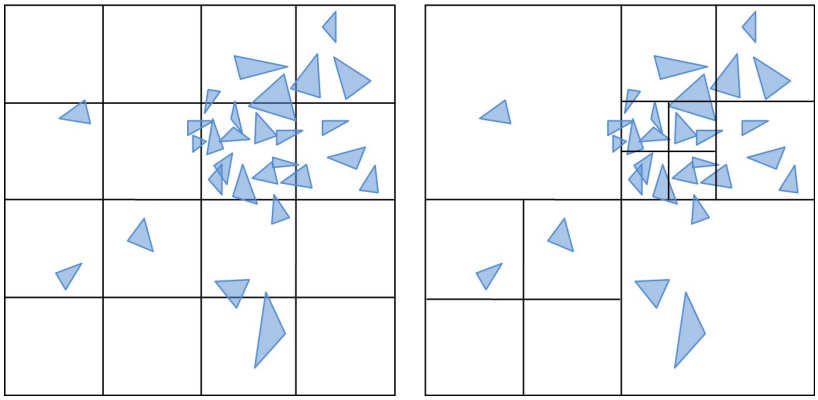


Figure 6.1: A scene with uneven triangle spatial distribution is partitioned by (a) a uniform subdivision technique and (b) a non-uniform subdivision technique

the issue of uneven spatial distributions. Fan et al. [2011] partition dense cells, but their hierarchical grid only has two levels, so that the problem is just partially solved. Kroiss [2013] subdivides space into a hierarchy of multiple cell levels, then each primitive is assigned to a level whose cell size fits well. The hierarchical grid adopted by these methods can be seen as a full octree, which lacks the adaptability to reduce the number of tests as much as possible.

Octrees are a more beneficial approach to this problem, because the tree offers a better adaptation to distributions. In fact octrees are commonly used in CPU-based algorithms. Unfortunately, parallel construction, update and traversal of traditional trees on GPUs require complicated operations [Lauterbach et al., 2009; 2010], giving relatively low occupancy. Therefore it is worth investigating a GPU-based algorithm that adaptively partitions space, if the cost to achieve it is low.

Motivated by these limitations, we investigate an adaptive *octree grid* (OTG) to accelerate CD for deformable objects on a GPU. The OTG takes advantages of both octrees and hierarchical grids. The octree that adaptively subdivides space based on triangle distribution is efficiently represented on a GPU by a grid which is actually an array of a length equal to the number of cells of the hierarchical grid (full octree). Using the OTG yields a considerable decrease in the number of broad phase tests. Thus a *two-stage* scheme consisting of a *bottom-up* stage and a *top-down* stage is used to cheaply improve octree subdivision, minimising the number of tests. Additionally, the number of levels in the OTG is dynamically adjusted according to triangle spatial distribution instead of being fixed. Another benefit

arising intrinsically from this method is that uneven triangle sizes are also handled well, because uneven triangle sizes can be seen as uneven spatial distributions most of time. For example, in a given scene the triangle density of areas having refined meshes is very likely higher than that of areas having coarse meshes.

We use the CUDA toolkit 5.5 and have conducted experiments with a set of scenes using an NVIDIA GTX 780 GPU to investigate and analyse performance. The method can perform discrete (DCD) and continuous collision detection (CCD) of objects consisting of tens of thousands arbitrarily distributed triangles in a few milliseconds. Results show that the approach efficiently addresses issues of uneven triangle sizes and uneven triangle spatial distributions.

6.2 Related Work

For literature review of CD we refer the reader to Chapter 2. This section briefly reviews previous work most directly related to the work of this chapter.

Zhang and Yuen [2000] and Teschner et al. [2003] introduce a technique employing spatial hashing with a uniform grid for CD in simulation of deformable objects. Eitz and Lixu [Eitz and Lixu, 2007] extend it to a method with a hierarchical grid. The focus in recent years has shifted to GPU computing for CD. Grand [2007] describes how to use a uniform grid to perform broad phase CD on GPU with CUDA for particle systems. Later, [Pabst et al., 2010] extend this method to handle CD for deformable triangular meshes. To improve performance, Kroiss [2013] partially addresses the problem of uneven triangle sizes by a hierarchical grid. The previous chapter better handles this issue using a virtual subdivision scheme (VSS) with a uniform grid [Wong et al., 2012]. As mentioned above, the issue of uneven triangle spatial distributions is solved by [Fan et al., 2011; Kroiss, 2013] in part.

6.3 Approach

Like many existing efficient CD algorithms, our CD pipeline consists of a broad phase and a narrow phase. The task of the former is to cheaply prune unnecessary tests for triangle and other elementary pairs, such as vertex-triangle (VT) and edge-edge (EE) pairs, that are far away from each other. In the narrow phase, proximity tests are performed to compute exact intersection information of elementary pairs, which are also referred to as elementary tests. AABBs and K-DOPs (specifically 18-DOPs) are used as the BV for the broad phase and

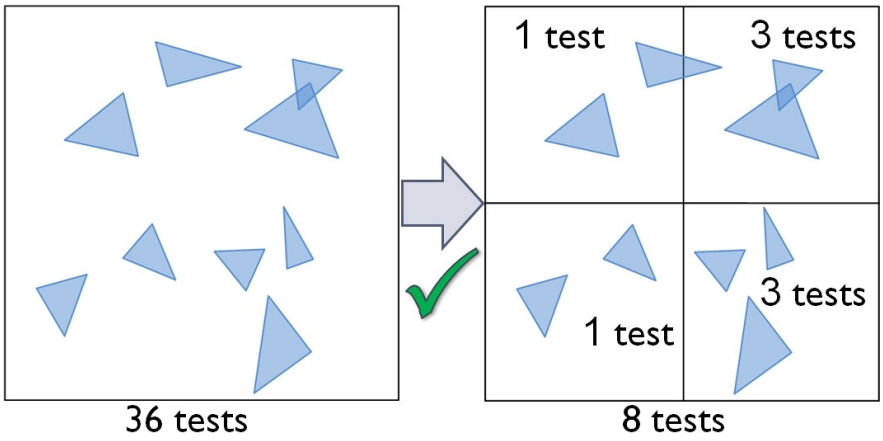


Figure 6.2: The subdivision reduces the number of tests.

narrow phase respectively in our work. Contributions of the OTG method relate primarily to the broad phase.

6.3.1 Adaptive Space Partition

As mentioned earlier, uneven triangle distributions can make spatial subdivision very inefficient. Therefore adaptively partitioning space according to distribution densities can offer advantages. We use an OTG to represent the adaptive subdivision, with triangles ultimately assigned to the leaf cells determined not to be subdivided. The top (coarsest) level is just a single cell which is defined as the AABB of the scene at the current time step. Cell sizes of lower levels are computed by halving the cell size of the parent level in x, y and z directions. Now a criteria to determine whether a cell should be subdivided is needed.

Since CD tests are performed only for triangle pairs in the same cell, the number of tests of a cell containing n triangles is $n(n-1)/2$. It is reasonable to define a simple criteria that a cell needs to be subdivided if the number of tests decreases after subdivision. For example, for the 2D case shown in Figure 6.2, subdivision reduces the number of tests from 36 to 8. In other words, a cell should not be subdivided if subdivision increases the number of tests (Figure 6.3). With this criteria, the OTG can be built using a single-stage scheme. For example, it can be constructed top-down from the root to leaves to subdivide cells recursively until the criteria of reducing the number of tests fails. Alternatively, a bottom-up scheme can be used to construct the OTG. The OTG constructed by a single-stage scheme is referred to as OTG_s.

Although an OTG_s offers a much better adaptation to spatial distributions compared to

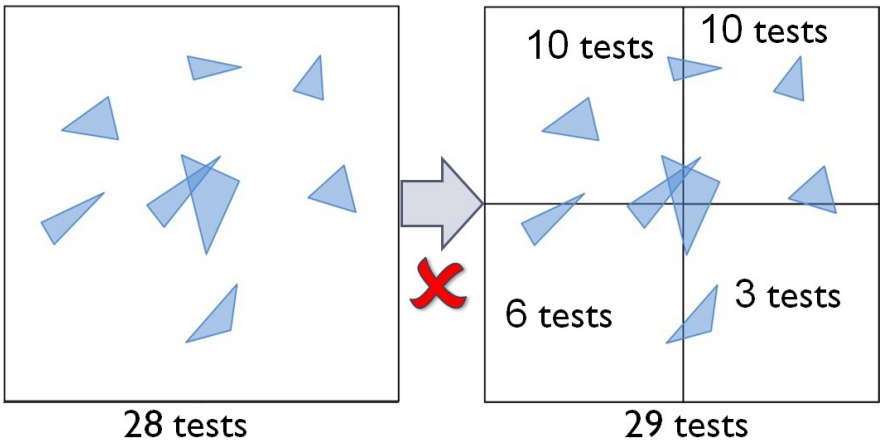


Figure 6.3: The subdivision increases the number of tests.

previous CD methods on GPUs, the efficiency of octree subdivision can be further improved. In the example shown in Figure 6.4, the first subdivision increases the number of tests, so subdivision would stop with the simple criteria, although further subdividing three of the cells significantly reduces the number of tests. Hence deciding when to stop requires recording and comparing to other alternative subdivisions. This can dramatically increase the complexity since the number of subdivision combinations grows exponentially as the number of levels increases. A bottom-up scheme that determines if sub-cells need to be merged suffers the same limitations (Figure 6.5). Hence a scheme that cheaply generates more efficient partitioning is required. We achieve this by using a two-stage scheme to construct the OTG, which is referred to as OTG_t.

The OTG_t is built as follows. At first a counter octree and a type octree are built to record the triangle distribution information and cell types (internal, leaf or inactive). The counter tree is actually a hierarchy table stored as an array of a length equal to the number of the hierarchical grid (full octree) cells, in which each element corresponds to a cell in the hierarchy and records the number of triangles overlapped by this cell. Grid cells are marked as internal, leaf or inactive cells to represent the octree, therefore traditional operations on tress such as adding, deleting and pruning are not needed, and thus the array length does not change during simulation. The type tree is treated as the same to store cell types. In fact on GPUs, maintaining and managing trees in this way is significantly more efficient in time than the traditional way of operating a tree, albeit at the cost of memory, which we found not to be a constraint in this work.

Next, all triangles are temporarily assigned to the base (finest) subdivision level cells only.

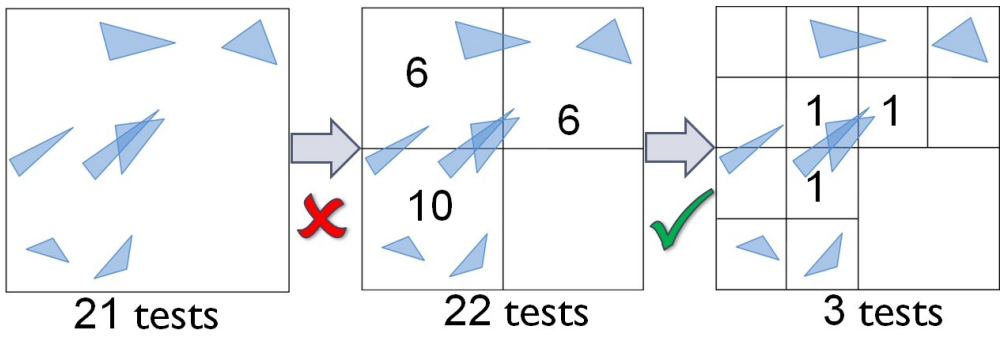


Figure 6.4: The limitation of a simple top-down scheme.

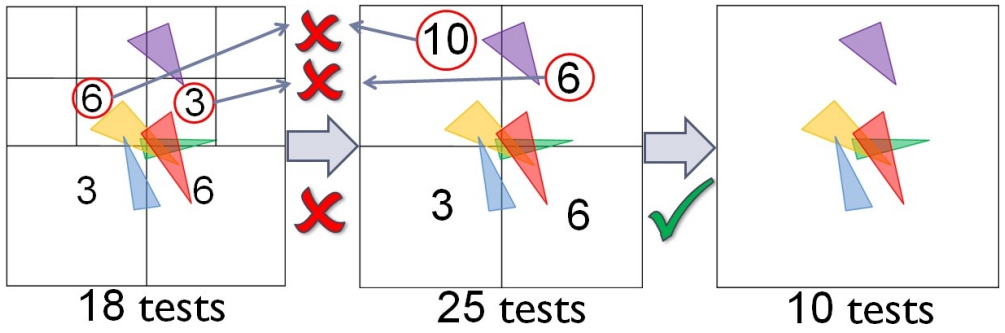


Figure 6.5: A simple bottom-up scheme suffers a similar limitation.

Since triangles are very unlikely to overlap similar number of cells, this assignment step is performed using the workload distribution scheme proposed by [Fan et al., 2011]. During this process the number of triangles assigned to each base level cell is obtained using atomic operations. For the number of triangles overlapped by higher subdivision level cells, we only cheaply record the number in the counter tree by mapping triangles to higher cells instead of expensively assigning them to higher cells. This is done by employing a token position technique [Fan et al., 2011] which efficiently removes the redundancy caused by the fact that a triangle assigned to multiple base cells can map to the same higher cell. Once the counter tree has distribution information, empty cells are marked as inactive cells, and the number of triangles overlapping each cell is converted to the number of tests.

Then, a two-stage scheme is used to obtain an improved octree subdivision. At first a bottom-up stage, level by level, checks if all eight child cells of each parent cell should be merged or not using the simple criteria, starting from the base level. If they should the only thing required to do is to mark the parent cell as a leaf cell. Otherwise, the eight child cells are marked as leaf cells and the parent cell marked as an internal cell. Unlike the simple single

stage scheme, the bottom-up stage does not stop until the top level is reached. Therefore, in the counter tree, the number of tests of the internal parent cell should be replaced with the sum of numbers of tests of its eight child cells. After this stage is done, the highest level leaf cell in a branch is taken as the real leaf cell.

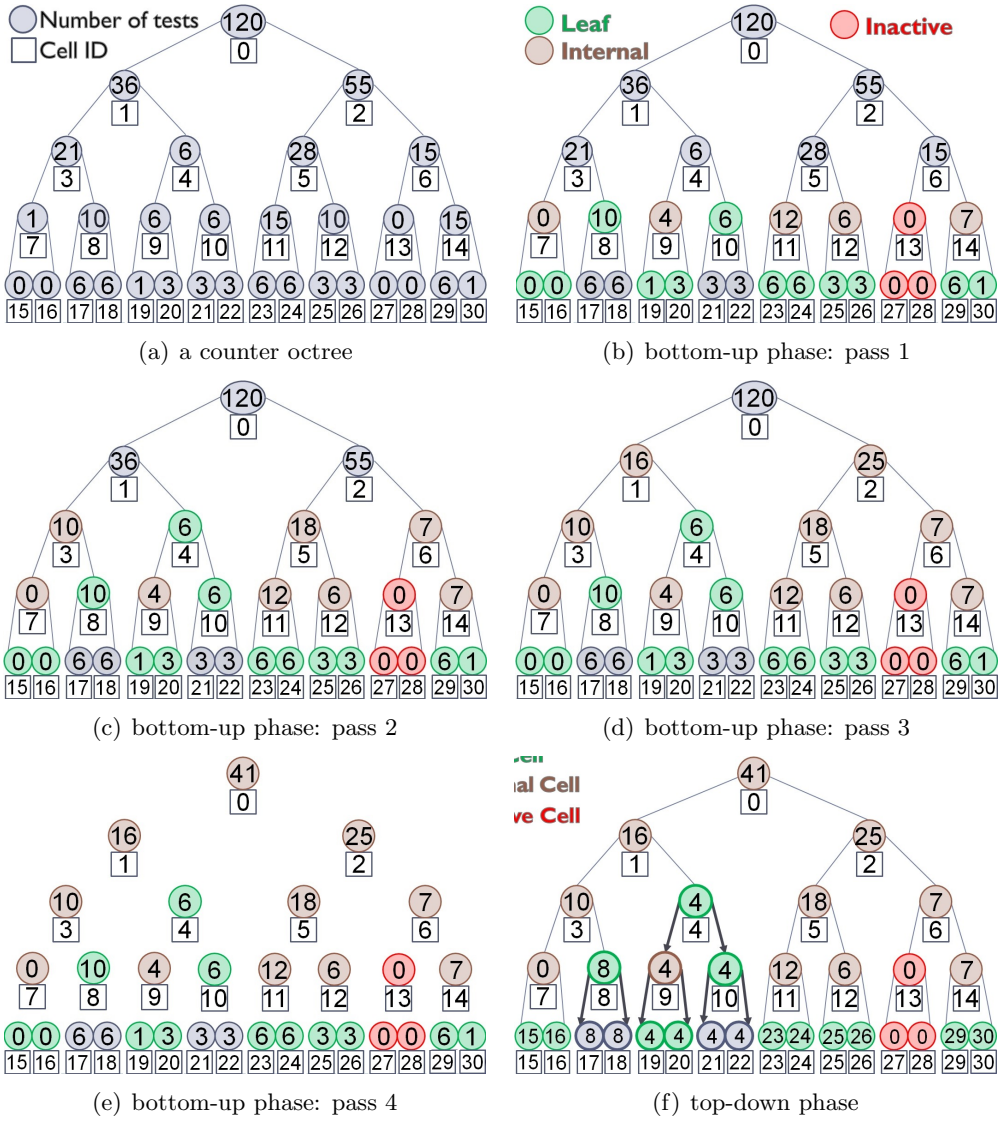
In the top-down stage, the first step is for all leaf cells to replace the number of tests stored in the counter tree with the cell identifier (ID). Then IDs of the highest leaf cells are propagated down to elements corresponding to base level cells (excluding inactive cells), so the cells that triangles should be finally assigned to can be easily obtained by accessing base level elements of the counter tree. Now the OTG_t that better subdivides space is built.

These two stages are implemented in two separate CUDA kernels that traverse the hierarchy level by level. The number of invoked threads in each iteration equals the number of cells in the corresponding level.

A 1D example is given in Figure 6.6. The two-stage scheme easily removes the limitation of single stage schemes. Although cells 19, 20 and 10 are marked as leaf cells in pass 1, the bottom-up stage does not stop and cell 4 is marked as a leaf in pass 2 as well. The top-down stage brings down IDs of the highest leaf cells to their child cells, which corrects the determination made in pass 1. An optimised subdivision that reduces the number of tests from 120 to 41 is obtained.

6.3.2 The Number of Cell Levels

Since building the OTG starts with a bottom-up stage, the number of levels needs to be determined. Instead of using a hierarchy with a fixed number of levels, we dynamically adjust it during the simulation after the first time step. We start with a grid of nine levels in the first time step. After the two-stage scheme is done, how many triangles are assigned to each level is known. In terms of efficiency, it would be better to avoid calculations for inactive levels which are levels without any triangle assigned. Therefore the base and top level in the next time step can be determined from the levels in the current time step. Assuming that the highest and lowest active levels in the current step are the ith and jth level ($i < j$) respectively, the top level in the next step is set as the $(i-1)$th level if triangles assigned to the highest active level in the current step are more than a customised threshold $\epsilon_{\text{-}}$ (empirically set to 0.1%), or it is set as the $(i+1)$th level if triangles are less than a threshold $\epsilon_{\text{+}}$ (0.001%), otherwise it remains as the ith level. Similarly, the base level in the next step is set as the $(j+1)$th level if triangles assigned to the lowest active level in the current step are more than



(a) a counter octree (b) bottom-up phase: pass 1

(c) bottom-up phase: pass 2 (d) bottom-up phase: pass 3

(e) bottom-up phase: pass 4 (f) top-down phase

Figure 6.6: A 1D example that each cell can be subdivided into two child cells is shown. In this counter octree, the number in a box represents a cell ID, and the number in a circle indicates the number of tests in that cell. (a) After the counter octree which records the distribution information is built. The bottom-up stage is performed starting from the base level to check if all child cells (two cells in the 1D case) of every parent cell should be merged or not, according to the simple criteria. (b) During pass 1*: cell* $8, 10, 15, 16, 19, 20, 23, 24, 25, 26, 29$ *and* 30 *marked as leaf (green) cells, cell* $7, 9, 11, 12$ *and* 14 *are marked as normal cells. For those parent cells that are normal, the number of tests is replaced with the sum of numbers of tests of its child cells. (c, d, and e) This step is performed iteratively on upper levels until the top level is processed. Note that the first iteration processes two levels. (f) Finally, an top-down stage brings IDs highest leaf cells to their child cells level by level until the base level is reached, which corrects determinations made in previous passes.*

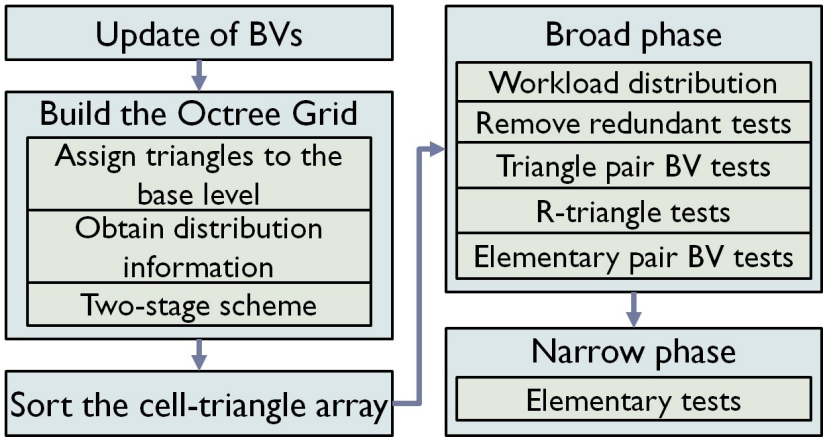


Figure 6.7: The pipeline of the CD algorithm.

a threshold λ_+ (97%), or it is set as the $(j-1)$th level if triangles are less than a threshold λ_- (3%), otherwise it remains as the jth level. The two-stage scheme is only performed for levels between the top and base levels.

Intuitively, it seems that incrementally changing the octree by directly assigning triangles to leaf cells of previous step rather than rebuilding the tree from the hierarchical grid by assigning them to the base level of previous step might be more efficient. Unfortunately a triangle can belong to multiple levels, so levels to which each triangle belongs have to be recorded for the next step, and for each triangle the assignment step may need to be performed multiple times depending on how many levels the triangle belong to, leading to an increase in costs for incremental update. Moreover since one thread handles the assignment of one triangle, uneven workloads can further harm the performance because how many times the assignment step is performed in every thread can be different. Therefore we do not assign triangles to leaf cells directly.

This dynamic adjustment mechanism works well in our benchmarks. Active levels are between the fourth and seventh levels for most of the benchmarks. In fact, levels higher than the third level and levels lower than the eighth level are inactive for all benchmarks.

6.3.3 Remainder of The CD Pipeline

Once the space subdivision step is completed, a cell-triangle pair array is easily built, and then sorted by cell ID using the parallel radix sort algorithm [Harris, 2007].

In the broad phase, a workload distribution scheme presented in the previous chapter that assigns one broad phase test to a thread is adopted to balance computation load among

(a) Funnel (b) Cloth on ball (c) Flamenco (d) N-body

Figure 6.8: Benchmarks used for performance measuring.

concurrent threads. Then, in each thread the token position technique is employed to determine if the test of this thread is redundant, since a triangle pair can overlap multiple cells. Although the more efficient control bits scheme is not used for this step because more than eight cells can be overlapped by a triangle (refer to Chapter 3 for detail), the reduction in the number of broad phase tests yielded by the OTG is remarkable enough to improve overall performance. Next, a BV test for the triangle pair is performed. If the BVs intersect, R-triangle tests that remove redundant elementary tests are carried out. The last step of the broad phase is to perform a BV test for each elementary pair to further improve the culling efficiency. The collision pipeline up to here yields a considerable reduction in the number of elementary tests.

In the narrow phase which is the same to the VSS method, proximity is checked for VT and EE pairs output by the broad phase to compute exact intersection separately by launching two different kernels to avoid low occupancy. Before the proximity test, a cubic equation needs to be solved to compute the first contact time in the case of CCD [Provot, 1997]. Figure 6.7 shows the overview of the collision pipeline.

6.4 Implementation and Results

The method is implemented using the CUDA toolkit 5.5 on a computer with an NVIDIA GTX 780 GPU. The method is purely GPU-based, and data transfer between CPU and GPU

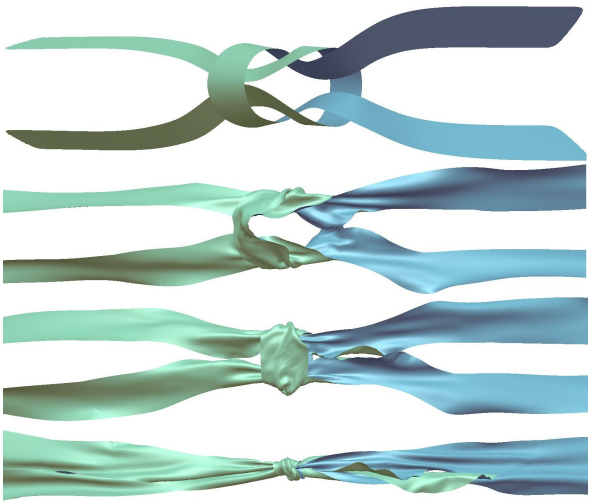


Figure 6.9: Two pieces of ribbon are tied into a reef knot.

is performed only once at the simulation beginning, so that performance completely depends on the GPU. We also run the method on a GTX 470 GPU for comparison with previously reported methods.

6.4.1 Benchmarks

Same experiments with the previous chapter are used for measurement for this work. Motion data of the *cloth on ball*, *flamenco* and *N-body* tests were provided by the UNC Dynamic Scene Benchmarks collection. The spring-mass method is used to compute cloth internal forces with an implicit integrator (Equations 3.25, 3.26 and 3.27) for other benchmarks.

1. Funnel (20K triangles): A piece of cloth falls into a narrow funnel and goes through it. (Figure 6.8(a))

2. Cloth on ball (92K triangles): A cloth with a large number of triangles drapes on a sphere, which starts rotating. (Figure 6.8(b))

3. Flamenco (49K triangles): A dancing character wearing a skirt with multiple layers of cloth. (Figure 6.8(c))

4. N-body (146K triangles): Hundreds of spheres and five cones are colliding with each other. (Figure 6.8(d))

		Level					
		3	4	5	6	7	8
Funnel	A.	-	100	100	100	23.6	-
	R.	-	4.7e-3	2.1e-1	70.8	29.1	-
Cloth ball	A.	1.6	100	100	100	87.5	3.4
	R.	1.6e-3	5.4e-2	3.5e-1	2.2	92.7	4.9
Flamenco	A.	4.9	100	100	100	87.2	-
	R.	1.5e-3	6.1e-2	5.4e-1	43.6	56.3	-
N-body	A.	12.5	100	100	100	96.0	10.1
	R.	2.9e-3	8.3e-2	4.7e-1	6.5	80.2	13.3
Reef knot	A.	-	100	100	100	75.3	-
	R.	-	4.5e-2	2.0e-1	10.7	88.3	-

Table 6.1: The ratio (in percentage) between the number of time steps in which each level is active (A) and the number of total time steps, and the ratio (in percentage) between the number of triangle-cell registrations (R) in each level and the number of the total registrations are given.

5. Reef knot (20K triangles): Two pieces of ribbon are tied into a reef knot, resulting in a very uneven triangle distribution. (Figure 6.9)

The triangle spatial distribution density in these benchmarks is uneven in most time steps. For example, in the *cloth on ball* and *reef knot* benchmarks, triangle distributions of areas under the sphere and near the knot are very dense, while triangle distributions of other areas are relatively sparse. Similarly in the *flamenco* test, the triangle density of the skirt of multiple layers of cloth is higher than that of the shirt, because the skirt consists of multiple layers of cloth. In addition, some benchmarks consist of meshes whose triangle sizes vary considerably.

6.4.2 Results and Analysis

Active levels of the OTG are adjusted depending on the distribution of triangles in the hierarchy. In all benchmarks, the ratio between the number of frames in which each level is active and the number of total frames, and the ratio between the number of triangle-cell registrations in each level and the number of the total triangle-cell registrations, are given in Table 6.1. Levels 1, 2 and 9 are inactive during the whole simulation for all benchmarks. In contrast levels 4, 5, 6 and 7 are always active. Most triangle-cell pairs are registered in levels 6 and 7.

The key idea of the OTG method is to reduce the number of unnecessary broad phase tests resulting from an uneven triangle spatial distribution. We compare the average number of

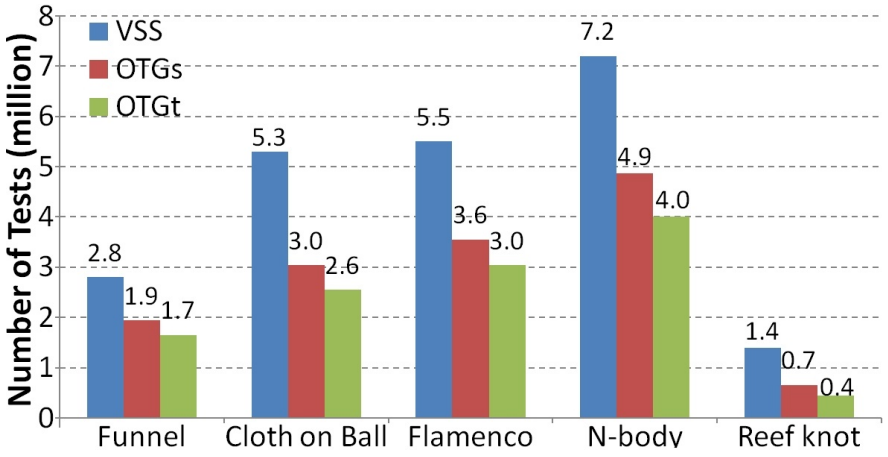


Figure 6.10: Comparison of the average number of broad phase tests in CCD among the VSS method, the OTG_s method and the OTG_t method.

broad phase tests per frame between the OTG_t method and the VSS method which drastically reduces the broad phase tests for a uniform subdivision. As Figure 6.10 shows, the OTG_t method yields a significant reduction in tests, by at least 39%, across all benchmarks. The largest reduction, of 64%, is observed in the *reef knot* test. This is because the density of the knot area is very high and the density of remaining areas is low, resulting in an inefficient subdivision for a uniform grid as expected. In contrast the OTG_t adaptively partitions the space according to the distribution density.

In order to demonstrate the further improvement of OTG_t, the number of broad phase tests produced by the OTG_s method is also given in this figure. In all benchmarks, the OTG_t method partitions space for a better adaptation to triangle distribution, yielding less tests compared to OTG_s by up to 36%, and an average of 20%.

A common feature of CD methods employing space subdivision techniques is the large memory requirement, because of the large number of broad phase tests. This feature becomes a major limitation for GPU-based methods. However, memory usage for tests is not our main concern, because the two-stage scheme is designed to generate as few as possible broad phase tests. However, memory is required for two octrees. Maintaining a tree grid as a table requires a relatively large amount of memory. The memory size of the counter octree of nine levels is 512MB, and the size of the type octree of nine levels is 64MB because the base level is unnecessary (the type of base level cells can be obtained from their parent cells). Fortunately, for the test scenes this is not a limitation in an environment with a GTX 780 GPU whose memory is 3GB. The number of OTG levels for all these standard benchmarks never reaches

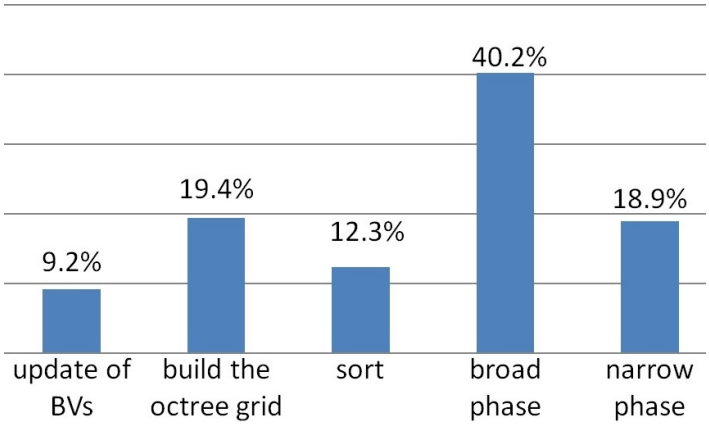


Figure 6.11: Time proportions of each step for cloth on ball benchmark

nine, which means a memory of 72MB satisfies two octrees. For extreme situations, spatial hashing can be adopted to easily avoid using more than nine levels.

6.4.3 Performance

The performance of the OTG$_t$ method for DCD and CCD on benchmarks is given in Table 6.2. The results demonstrate that this method works well with scenes with uneven triangle spatial distributions. Execution times of CCD on steps for *cloth on ball* are presented in Figure 6.11. The most expensive part is the broad phase which takes 40%, while computation time of building the OTG takes 19% of the total running time.

6.4.4 Comparison

We first compare the OTG$_t$ method against the VSS method, not only because the VSS method provides the fastest results for most benchmarks among previous methods we are aware of, but also the OTG$_t$ method is designed to address the uneven triangle spatial distribution issue, the main limitation of the VSS method. In order to remove uncertain factors arising from different hardware, the VSS method is run on the experiment environment used in this chapter. Performance of the VSS method measured on a GTX 780 GPU is given in Table 6.3 to provide a straightforward comparison.

Performance of the OTG$_t$ method for CCD on all benchmarks is faster with speedups of 1.5X and 1.3X for the *reef knot* and *N-body* benchmarks. For DCD, the VSS method gives slightly better performance for the *flamenco* and *n-body* benchmarks, while the OTG$_t$ method is around 1.3X, 1.2X and 1.2X faster for the *reef knot*, *cloth on ball* and *funnel*

		GTX 470		GTX 780	
	Triangles × 1000	DCD (ms)	CCD (ms)	DCD (ms)	CCD (ms)
Funnel	18	4.7	5.1	3.7	3.4
Cloth ball	92	10.5	13.3	8.3	9.2
Flamenco	49	16.1	17.4	11.7	10.8
N-body	146	30.2	24.8	20.8	19.1
Reef knot	20	4.7	4.9	3.2	3.0

Table 6.2: Timings of the OTG_t method for DCD and CCD, measured on a GTX 470 GPU and a GTX 780 GPU.

	DCD (ms)	Speedup (times)	CCD (ms)	Speedup (times)
Funnel	4.4	1.2	4.0	1.2
Cloth ball	10.2	1.2	11.1	1.2
Flamenco	11.2	0.9	11.0	1.0
N-body	18.7	0.9	24.5	1.3
Reef knot	4.3	1.3	4.4	1.5

Table 6.3: The DCD and CCD timings of the VSS method run on the experimental environment with a GTX 780 GPU, and speedups of the OTG_t method.

tests respectively. Although these benchmarks consist of triangles whose size also varies significantly and the VSS method is designed to handle this problem, the OTG_t method is superior for the VSS method for most tests, which demonstrates that the OTG is able to address the uneven triangle size issue as well. However the improvement in performance is not as remarkable as the significant reduction in the number of broad phase tests, and even not observed for two of the DCD tests. This is mainly because the control bits scheme employed by the VSS method to remove the redundancy arising from the fact that a triangle pair can overlap multiple cells is more efficient than the token position technique used in the OTG approach, which cancels out part of the performance gain. It is also noticeable that the increase in CCD performance is greater than that in DCD. The reason is that in our cloth simulation framework, by following [Bridson et al., 2002], the BV is enlarged by a cloth thickness in the DCD case, while in the CCD case the BV is enlarged by a error tolerance which is much smaller than the cloth thickness. Therefore BVs of triangle pairs are more likely to overlap in the case of DCD, thus there is less potential for the OTG to reduce the number of tests. This is also why the OTG_t method is faster for CCD of the *n-body* benchmark, but slower for DCD.

Fan et al. [2011] address the issue of uneven spatial distribution in part by presenting a two level hierarchical grid. Their reported performance results are measured on a GTX 480 GPU, whilst we compare with results measured on a slightly slower GTX 470 GPU. The OTG_t method is faster for CCD in all except for the *flamenco* benchmark. For the *cloth on ball*, *funnel* and *n-body* tests, speedups of from 1.3X to 1.9X are observed. We do not provide a comparison of DCD performance, because for DCD in the narrow phase they only check collisions between triangle pairs, while we perform collision tests on VT and EE pairs to compute exact collision and contact information.

To the best of our knowledge, Tang et al. [2011] report the fastest results among GPU-based methods employing BVHs. They also run experiments on a GTX 480 GPU. CCD performance of the OTG_t method measured on a GTX 470 GPU is better in *cloth on ball*, *flamenco* and *n-body* benchmarks: 13.3ms vs. 18.6ms, 17.4ms vs. 32.7ms and 24.8ms vs. 79.0ms, respectively, although is slightly slower for the *funnel* test.

Finally, by comparing our CCD timings on a GTX 470 GPU to the fastest multicore CPU-based method [Tang et al., 2010b] where experiments are run on a PC with four quad-core CPUs (16 cores) at 2.93 GHz, speedups of 1.6X and 1.3X are observed for the *flamenco* and *cloth on ball* tests, on which the multicore CPU-based method method takes 27ms and 32.5ms. Performance of the OTG_t method on a GTX 780 GPU for these two tests are 2.5X and 1.7X faster. DCD timings of these methods are not provided.

6.5 Conclusion

This chapter presents a GPU-based method using an OTG to accelerate CD of deformable objects by adaptively subdividing space according to triangle distribution. The OTG is dynamically built and adjusted. A two-stage scheme is introduced to optimise octree spatial subdivision and reduce the number of broad phase tests. The method addresses the issue of uneven triangle spatial distributions, a common limitation not addressed well in previous spatial subdivision based CD approaches on GPUs. Further, the uneven triangle size problem can be addressed by this method as well, since it can be viewed as a subcase of uneven spatial distributions. We investigated the method by conducting a set of standard benchmarks, and observed increases in performance compared to previous CPU-based and GPU-based approaches.

Chapter 7

An Improved Friction Model

Once all collisions are detected, they have to be dealt with. Friction is an important aspect of this process. It is a complex phenomenon that resists tangential motion between two contacting objects. In computer graphics, existing cloth simulation methods have typically used a Coulomb model for friction, both static and kinetic, as discussed in Chapter 3. However, the Coulomb model is limited since it intends to capture fundamental frictional phenomena. Many advanced friction effects such as stiction, Stribeck friction, viscous friction and the stick-slip phenomenon are not modelled by it. This chapter describes an improved physically based friction model for simulating such effects, extending the Coulomb model. We show that including these additional effects is relatively straightforward. Results and comparisons with a pure Coulomb model are provided to demonstrate the capabilities and features of the improved model.

7.1 Introduction

Friction is an important and complex phenomenon which occurs whenever two objects are in contact in nature. Its development is essential in tribology. Since Leonardo da Vinci and Amonton discovered rules of dry friction, it has been studied in many research fields such as mechanical engineering, material science, fluid and solid object dynamics, and so on. It also plays a significant role in fabric performance, from textile processing to user's tactile comfort. Over time a number of friction models have been developed to capture more observed features. In computer graphics, as an indispensable component of collision handling in physics simulation, friction greatly affects both static and dynamical behaviour. Existing cloth, hair and solid object simulations in graphics typically use a Coulomb model. A more

comprehensive friction model is worth investigating.

When two bodies are in contact, dry friction occurs at the contact area as a force resisting the relative tangential motion between the contacting surfaces. Dry friction is separated into static (stiction) and kinetic friction for motionless and sliding contacts respectively, and can be described by three classic empirical laws:

- Amontons' 1st Law: friction is directly proportional to the applied normal direction.
- Amontons' 2nd Law: friction is independent of obvious contact area.
- Coulomb's Law: kinetic friction is independent of the sliding velocity.

Friction described by these laws is termed Coulomb friction. Because of its simplicity it has been widely used to model friction behaviour. As investigation into friction continued, many phenomena that could not be fully modelled by classic laws were observed, leading to more elaborate models extending the Coulomb model.

One of the most important phenomena not captured by the Coulomb model is that the break-away force (stiction) required to initiate sliding motion from rest is usually larger than kinetic friction [Morin, 1833]. This is observed for almost all contact surfaces. The difference between stitction and kinetic friction causes intermittent motion rather than smooth movement between two objects in contact at low relative speeds. Typically, two contacting objects will not start sliding until the applied force is larger than stiction. Then objects suddenly slide across each other under smaller kinetic friction for a short while until they stick again. This frequent phenomenon is referred to as the stick-slip phenomenon (simply stick-slip) [Taylor and Pollet, 2000].

Once the tangential force exceeds the stiction, friction transits from static to kinetic form where another two important phenomena occur, the Stribeck effect and viscosity resulting from the dependency of friction on the relative tangential velocity. The Stribeck effect is the decrease in friction from the stiction peak to lower Coulomb friction when the sliding velocity is small [Stribeck, 1902]. Viscosity indicates that friction increases as the sliding velocity increases [Reynolds, 1886]. In physics and mechanical engineering, models which are able to capture these phenomena have been developed and successfully used to model friction for years.

These phenomena have been mentioned in discussion of friction of fabrics and fibres [Das et al., 2005; Ajayi, 1992; Fetfatsidis et al., 2009]. Fabrics exhibit the feature of stiction larger than the kinetic friction resulting from its structural unflattening. According to analysis by

Taylor and Pollet [Taylor and Pollet, 2000], if the fabric has a certain degree of elasticity, stick-slip occurs at low relative speeds due to the difference between higher stiction and lower kinetic friction. Solid objects such as metals show the Stribeck and viscous friction only under moist or wet conditions, while such velocity-dependent frictional effects are observed under both dry and moist conditions for textile materials, i.e. fibre, yarns and fabrics that deform viscoelastically [Kalyanaraman, 1988; Ajayi, 1992; Fetfatsidis et al., 2009]. These effects have been thoroughly investigated for dry and lubricated textile materials.

In this chapter we describe an improved physically based friction model in cloth simulation that extends the Coulomb model to consider these friction phenomena, and report differences in overall cloth behaviour. The model can be easily integrated into most existing cloth systems as a replacement for Coulomb friction, increasing the physical plausibility by accounting for more friction effects. The model is not much more complicated than the Coulomb model and adds little extra computational cost overall. Several experiments and comparisons with cloth simulation using Coulomb friction and the extended models are provided.

The remainder of the chapter is organised as follows. We start by briefly reviewing previous work directly related to dry friction from engineering research and computer graphics in Section 7.2. The extended friction model is presented in Section 7.3, followed by discussion of implementation, results and performance in Section 7.4, 7.6 and 7.5 respectively. Finally, the conclusion and future work are given in Section 7.7.

7.2 Related Work

Friction is computed in collision response, for which we refer reader to Chapter 3. We provide a brief introduction to friction in engineering research, then previous work directly related to friction in computer graphics is reviewed.

Early friction models were based on classic laws which only summarise basic aspects of friction. Lately, requirements on more precise estimates of friction behaviour have become the main interest. A number of works [Stribeck, 1902; Karnopp, 1985; Armstrong-Hélouvry et al., 1994] improve on the Coulomb model to take into account effects discussed above for a wide range of applications. Their focus is on static models, on which our method is based. For purposes in the microscopic level in mechanical engineering, dynamic models such as Dahl, Bliman-sorine and LuGre models factor in evolution of friction through time or displacement derivatives, which however is beyond the scope of this work. We refer interested readers

to [Altpeter, 1999; Olsson et al., 1998]. In textile research, investigation of both static and dynamic friction models for fabrics and fibres have been reported in [Das et al., 2005; Lahey, 2002]. Experimental data by Taylor and Pollet [2000] show that fabric materials clearly exhibit the stick-slip behaviour.

In computer graphics, the earliest work including friction for cloth is [Terzopoulos et al., 1987]. Baraff and Witkin [1998] treat cloth-object friction as constraints. Bridson et al. [2002] propose a robust framework that computes friction as impulses for cloth simulation. Based on this framework, Selle et al. [2009] propose a novel cloth-object contact and friction scheme to achieve more accurate collision response. Pabst et al. [2009] introduce an anisotropic friction model for contacts between deformable objects (including cloth) and materials with anisotropic surface and inhomogeneous roughness. Chen et al. [2013] develop an algorithm to improve cloth-body interactions using friction data captured from real-world. However, to the best of our knowledge, no attempt to model effects of stiction, Stribeck and viscous friction has yet been made for cloth simulation in computer graphics.

7.3 Friction Model

We assume that at the contact point (Figure 7.1) the infinitesimal surfaces of two bodies A and B are smooth, so that the relative velocity $\mathbf{v}$ can be decomposed into a component $\mathbf{v}_n$ along the normal direction and a tangential component $\mathbf{v}_t$ opposed to the sliding direction. In the vertex-triangle type of contact, the normal of the triangle face is the contact normal, while in the edge-edge type of contact, the vector connecting the pair of closest points on two edges is treated as the contact normal. In this chapter we use the notations as abbreviation for different friction models for convenience:

- C: Coulomb

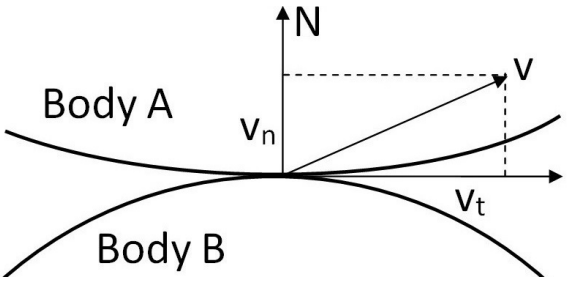
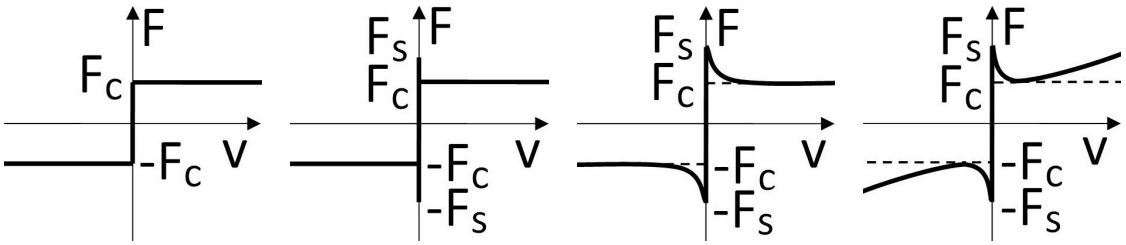


Figure 7.1: A contact between two bodies. The normal and tangential vectors are defined. The relative velocity is decomposed into the normal and tangential velocity.



(a) C (b) CS (c) CSS (d) CSSV

Figure 7.2: Friction models

- CS: Coulomb and sticition
- CSS: Coulomb, stiction and Stribeck friction
- CSSV: Coulomb, stiction, Stribeck and viscous friction

7.3.1 Friction Terms

This section describes each friction effect shown in Figure 7.2 in detail. The simplest model as well as the most common one in graphics is Coulomb friction (Figure 7.2(a)) as mentioned earlier. As described by three classic laws, the friction force $\mathbf{F}$ opposes the relative motion and is independent of the relative tangential velocity $\mathbf{v}_t$:

$$\begin{aligned} \mathbf{F} &= F_C(\text{-}\hat{\mathbf{v}}_t) \\ F_C &= \mu_c|\mathbf{F}_n| \end{aligned} \tag{7.1}$$

where F_C is Coulomb friction, μ_c the Coulomb coefficient, $\mathbf{F}_n$ the force in the normal direction of the contact plane and $\hat{\mathbf{v}}_t$ the unit vector of the relative tangential velocity.

Stiction describes the friction force at zero relative velocity, and as mentioned earlier stiction is usually higher than Coulomb friction (Figure 7.2(b)). Hence it is expressed as:

$$\mathbf{F} = \begin{cases} \text{-}\mathbf{F}_e & \text{if } \mathbf{v}_t = 0 \text{ and } |\mathbf{F}_e| < |F_S| \\ F_S(\text{-}\hat{\mathbf{F}}_e) & \text{if } \mathbf{v}_t = 0 \text{ and } |\mathbf{F}_e| \geq |F_S| \end{cases} \tag{7.2}$$

where $\mathbf{F}_e$ is the external tangential force, $\hat{\mathbf{F}}_e$ the unit vector of $\mathbf{F}_e$, and F_S the stiction threshold whose value is greater than or equal to Coulomb friction:

$$F_S = \mu_s|\mathbf{F}_n| \quad \mu_s \geq \mu_c \tag{7.3}$$

where μ_s is the coefficient of stiction. For fabrics μ_s is always larger than μ_c. The closer μ_s and μ_c are, the smoother the material is. For example, compared to silk, the stiction effect is more detectable in jeans. On the contrary the futher μ_s and μ_c are, the rougher the contacting surfaces are, more easily leading to stick-slip.

In order to model the Stribeck effect (Figure 7.2(c)) where friction decreases continuously from the stiction threshold to Coulomb friction as the velocity increases:

$$\mathbf{F}_{str}(\mathbf{v}_t) = (F_C + (F_S - F_C)e^{-(|\mathbf{v}_t|/v_S)^{\delta_S}})(\text{-}\hat{\mathbf{v}}_t) \tag{7.4}$$

where v_S is a positive value standing for the Stribeck velocity, and $\delta_S \in [0, 1]$ a curve shape factor describing the nonlinear decrease. Under dry conditions friction decreases faster than it does under moist conditions, but we only consider dry cases in this chapter. We set δ_S to 0.2 (the smaller the value, the more quickly friction decreases). In the Bliman-Sorine friction model [Bliman and Sorine, 1995b] Stribeck friction is degraded to be the velocity-independent transition from the stiction peak to Coulomb friction. This model has been adopted to compute internal friction in bending behaviour of fabrics [Lahey, 2002]. However either way removes the discontinuous decrease.

Under dry conditions, viscous friction in textile materials results from increasing frictional heating. For moist fabrics and lubricated fibre and yarns, viscous friction is characterized by the fact that a lubricant film exists between contact surfaces and friction is governed by the viscosity of the liquid. For either cases, viscous friction can be obtained by extending Coulomb model to consider the velocity as follows:

$$\mathbf{F} = F_v|\mathbf{v}_t|^{\delta_v}(\text{-}\hat{\mathbf{v}}_t) \tag{7.5}$$

where F_v is the viscous coefficient, and δ_v enables the nonlinearity of the dependency when its value is not 1. These two parameters vary a lot depending on types of textile materials in dry conditions, or types of fluid and how moist the condition is, i.e., moist to wet, in moist conditions. According to this equation, the higher the relative tangential velocity the larger friction is yielded (Figure 7.2(d)).

Therefore a general description of all effects is:

$$\mathbf{F}(\mathbf{v}_t) = \begin{cases} \mathbf{F}_k(\mathbf{v}_t) & \text{if } \mathbf{v}_t \neq 0 \\ \text{-}\mathbf{F}_e & \text{if } \mathbf{v}_t = 0 \text{ and } |\mathbf{F}_e| < |F_S| \\ F_S(\text{-}\hat{\mathbf{F}}_e) & \text{if } \mathbf{v}_t = 0 \text{ and } |\mathbf{F}_e| \geq |F_S| \end{cases} \tag{7.6}$$

where, $\mathbf{F}_k(\mathbf{v}_t)$ is the overall kinetic friction force as the sum of Coulomb, Stribeck and viscous friction:

$$\mathbf{F}_k(\mathbf{v}_t) = (F_C + (F_S - F_C)e^{-(|\mathbf{v}_t|/v_S)^{\delta_S}} + F_v|\mathbf{v}_t|^{\delta_v})(-\hat{\mathbf{v}}_t) \tag{7.7}$$

There are several contact methods for cloth simulation. The equations described above can be integrated into those systems to model different friction effects. In our work we discuss the use of the extended friction model in two commonly adopted contact methods: the constraint handling approach [Baraff and Witkin, 1998] and the velocity filter framework [Bridson et al., 2002]. The latter is described in Chapter 3. We use it for experiments in this chapter.

For constraint based models like [Baraff and Witkin, 1998], sticition and kinetic friction effects can be obtained by enforcing a complete constraint that eliminates relative motion and a partial constraint that only allows sliding on the plane proportional to the contact normal force, respectively. In the sticking case, the number of degrees of freedom for relative acceleration is set to 0, while in the sliding case the contact normal indicates the prohibited direction. The change in relative velocity caused by contact is included in the linear system for explicit control. The change in relative velocity is computed as $-\mathbf{v}_n + \mathbf{F}(\mathbf{v}_t)dt/m$, where m is the mass of the colliding primitive pair.

In order to replace the friction component in [Bridson et al., 2002] where friction effects are achieved by using impulses (Equation 3.37), friction needs to be converted into the impulse I, which can be simply computed as

$$I = \frac{1}{2}|\mathbf{F}(\mathbf{v}_t)\Delta t| \quad . \tag{7.8}$$

However in Equation 3.37 the impulse is directly obtained from the change in relative velocity in the normal direction $\Delta\mathbf{v}_n$ avoiding manipulating forces. Hence Equation 7.6 and 7.8 are replaced with:

$$\begin{aligned} \xi &= |\Delta\mathbf{v}_n|(\mu_c + (\mu_s - \mu_c)e^{-(|\mathbf{v}_t|/v_S)^{\delta_S}})/|\mathbf{v}_t| \\ &\quad +(F_v|\mathbf{v}_t|^{(\delta_v-1)})dt/m \\ I &= \begin{cases} \frac{1}{2}|\mathbf{v}_t| & \mu|\Delta\mathbf{v}_n|/|\mathbf{v}_t| \geq 1 \\ \\ \frac{1}{2}|\xi\mathbf{v}_t| & \mu|\Delta\mathbf{v}_n|/|\mathbf{v}_t| < 1 \end{cases} \end{aligned} \tag{7.9}$$

By applying the impulse I into impulse weighting equations 3.39 and 3.40, the relative tangential velocity will be cancelled out due to stiction or slowed down due to kinetic friction.

	incline plane	skirt	stick-slip	card	sphere
μ_c^{obj}	.2	.2	.1	.1	.2
μ_c^{self}	.1	.1	.1	.1	.1
μ_s^{obj}	.3	.4	.4	.2	.35
μ_s^{self}	.15	.2	.1	.2	.15
V_S	.4	.4	.4	.4	.4
δ_S	.2	.2	.2	.2	.2
F_v	N/A	N/A	N/A	.00006	.00006
δ_v	N/A	N/A	N/A	1	1

Table 7.1: Values of parameters used in each experiment. Superscripts obj and self indicate cloth-object and cloth self contacts respectively.

These two contact mechanisms discussed above are explicit, assuming the normal force and relative velocity are known. It is worth considering the extended friction model as part of implicit solvers (Equation 3.25), which we suggest as future work.

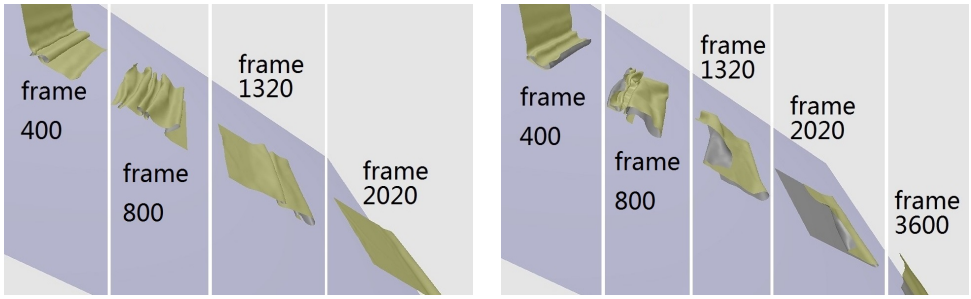
7.4 Implementation

We investigate this friction model by integrating which into a cloth framework presented in previous chapters. A model similar to [Baraff and Witkin, 1998] is used to model tensile forces with an implicit integrator (Equations 3.25, 3.28 and 3.29). Bending forces is computed using the model presented in Chapter 4, and integrated by an explicit integrator. Collision detection is handled in the way described in Chapter 6. For collision response the widely adopted framework of [Bridson et al., 2002] is used, but the friction model is replaced by the extended model discussed in this chapter. A strain rate limiting procedure similar to the scheme in [Bridson et al., 2002] is used to remove bad strain rates.

7.5 Results

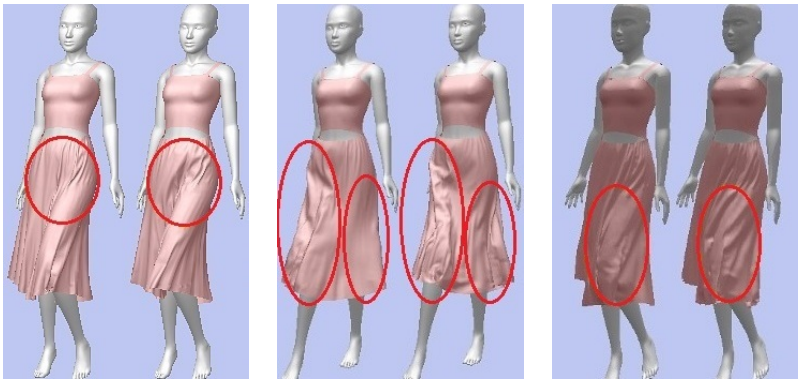
We investigate friction models using several scenarios, and point out differences and advantages by comparing with the C model. Friction is applied for both cloth-object contacts and self contacts. Since we are interested in differences in behaviour between different models, our aim is not to investigate and evaluate deeply how parameters of the extended model influence the output. Values of parameters used in each experiment are given in table 7.1. For overall dynamic behaviour we refer readers to accompanying animations.

We first test the CSS model. Figure 7.3 shows frames of a piece of cloth falling on an inclined plane. Figure 7.3(a) displays results of simulation using the C model, while Figure



(a) (b)

Figure 7.3: A piece of cloth falls on an incline plane. Results of (a) the C model and (b) the CSS model at several frames.

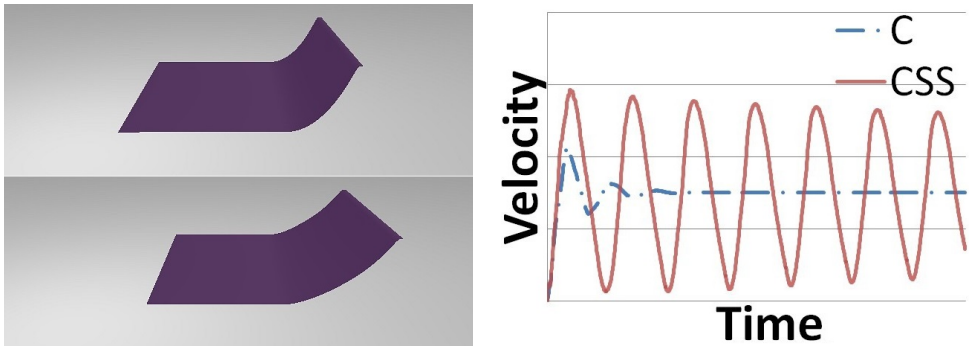


(a) (b) (c)

Figure 7.4: A walking character wearing a skirt. Results of the C (left) and CSS model (right) at several frames.

7.3(b) shows friction effects generated by the CSS model. Because of the higher stiction and the Stribeck effect, the cloth in simulation with the CSS model is more likely to experience static friction, and higher kinetic friction when the tangential velocity is small, and thus tends to fold more and slides slower than the cloth in the simulation using the C model. We have also applied these two friction models to a scene where a female wearing a skirt walks (Figure 7.4). In the CSS case, more and finer wrinkles are formed and hold longer, especially in the skirt as highlighted in the figure.

Since stiction larger than kinetic friction is the main cause of stick-slip, we built a scenario to test this mechanism (Figure 7.5(a)). A hang cloth has an edge attached. The lower part of the cloth touches the ground with friction. The attached edge moves at a constant velocity so that the cloth is pulled by the internal force. In the CSS model case where the kinetic friction



(a) (b)

Figure 7.5: (a) A scene to test stick-slip: results of the C (top) and CSS model (bottom). (b) Average displacements of the lower part cloth in simulations using the C and CSS model.

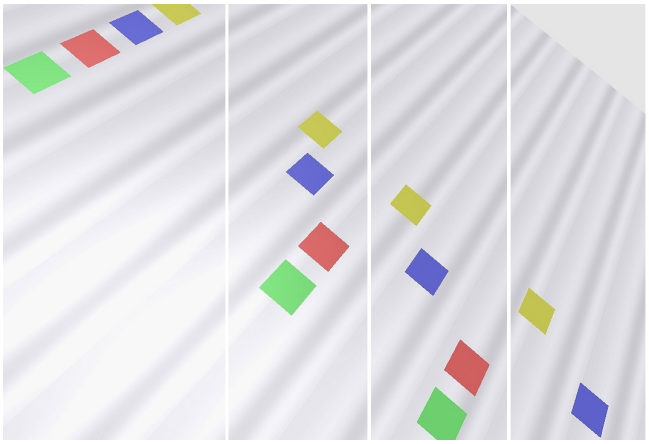
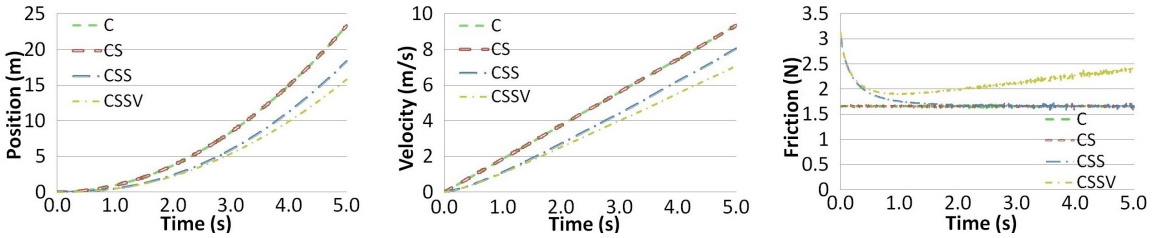


Figure 7.6: A card slides on an inclined plane. From left to right results of C model (green), CS model (red), CSS model (blue) and CSSV model (yellow) at several frames.

coefficient is much less than the stiction coefficient, the cloth moves intermittently. The cloth stays stuck before the tangential force reaches stiction, then it suddenly starts sliding with a large velocity due to the much smaller kinetic friction. In the slip-phase, the internal force decreases as the stretched cloth tends to recover to its unstretched state, leading to the cloth stuck again. As expected the entire process is repeated periodically. This stick-slip behaviour is depicted by the red solid curve in Figure 7.5(b), where average velocity of the lower part cloth are shown. On the contrary, the cloth in the C model case moves much smoother (the blue dashed curve) since there is no difference between stiction and kinetic friction.

Now we test all friction models. We build a test of four cards sliding on an inclined plane (Figure 7.6). Cards start from rest and accelerate. During the whole process, cards stay in

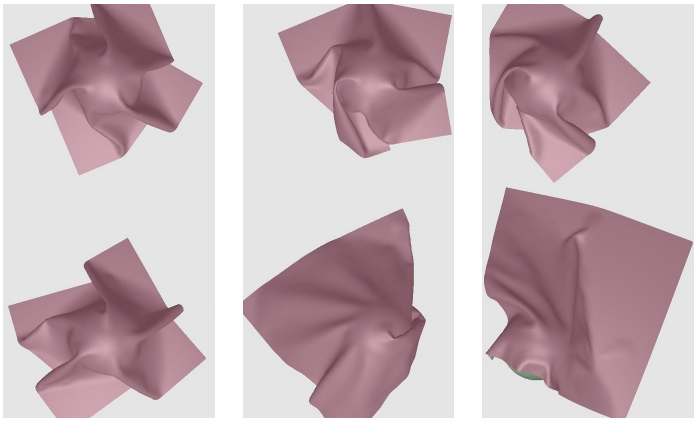


(a) Displacement (b) Velocity (c) Friction

Figure 7.7: The displacement, velocity and friction over time for a card sliding down on an incline plane using four different friction models are shown. Configurations of this test are identical to the cylinder experiment.

contact with the plane. All four friction models are used for four cards respectively. In all four cases, μ_s is set to $2\mu_c$. When the value of μ_c is above .3 no cards slide. If μ_c is set to between .15 and .3 only the card in simulation using the C model slides. By setting μ_c below .15, cards in all simulations move as the force towards the sliding direction is greater than the stiction, providing an initial sliding velocity. The value of μ_c in the experiment shown in Figure 7.6 is .1. Observed results match the expectation. Since the stiction cannot keep the card stuck, unsurprisingly the cloth in simulation using the C model has the same velocity with the one in the CS model simulation, as they undergo the same kinetic friction and therefore accelerate at an identical rate. However, the CSS and CSSV models yield higher kinetic friction at the beginning when the velocity is small, hence cards using these two models move slower. As their velocities increase, friction in the CSS model simulation gradually decreases to the Coulomb level, while friction by the CSSV model rises due to viscosity. Hence, the card using the CSSV model slides slowest. plots of friction, velocity and displacement over time for this experiment are shown in Figure 7.7.

Another scenario (Figure 7.8) in which a piece of cloth drapes over an accelerating rotating sphere is designed to show the effect of viscous friction in a more complicated case. As the angular speed of the sphere increases from zero, the cloth in the CSSV model case (bottom) spins faster and faster at an increasing acceleration due to increasing friction, while the cloth in simulation using the C model (top) spins at a constant acceleration caused by a constant friction force, as the friction is independent of velocity. Therefore, the cloth undergoing viscous friction is cast off earlier.



(a) (b) (c)

Figure 7.8: A cloth drapes over an accelerating rotating sphere. Results of the C model (top) and the CSSV model (bottom) at frame (a) 400, (b) 2400 and (c) 2980.

	no friction	C	CS	CSS	CSSV
time (s)	0.71	0.86	0.90	1.16	1.19
frition ratio	N/A	17%	21%	39%	40%

Table 7.2: For a simulation of 5 seconds ($\triangle t = 1ms$), running time of collision response using different models.

7.6 Performance

In our system computation time is dominated by collision detection, even though which is computed on a GPU. Collision response takes a small amount of total time. Hence the time for the friction part is even smaller. We use the scene (8.1K triangles) shown in Figure 7.3 to examine the performance. The time for collision response using different friction models are given in Table 7.2. The CSSV model introduces a 40% increase in collision response computation time which takes less than 8% of total time.

7.7 Conclusion

The extended friction model allows additional friction effects of stiction, including Stick-slip, Stribeck and viscosity easily incorporated into commonly adopted systems to replace the Coulomb model. Results of several experiments and comparisons are presented to demonstrate the capability of modelling friction effects and the resulting differences in cloth behaviour.

Chapter 8

Conclusion

In this thesis, four contributions to improve physically based cloth simulation have been considered. Chapter 4 reports our investigation into the use of a mathematical approximation to a hysteresis loop to effectively model non-linear effects in bending behaviour. Chapter 5 presents a GPU-based approach to improve performance of collision detection with a virtual subdivision scheme that better addresses the issue of uneven triangle sizes. Chapter 6 discusses a method that further improves collision detection performance by addressing both uneven triangle size and uneven spatial distribution issues using a GPU. Chapter 7 describes an extension to the Coulomb friction model to capture additional velocity-dependent frictional effects.

There are a variety of future directions for physically based cloth simulation. We mention some related to this thesis.

- **Further Aspects of Modelling Cloth Using Hysteresis**: The effects of hysteresis can be related to ageing and wearing, hence we would like to consider time as one factor influencing hysteresis in future work. In addition, a friendlier parametrization of the hysteresis controls can likely be developed. Hysteresis exhibited in cloth in-plane deformation is also worth investigating.
- **Further Improvements to Performance of GPU-based Collision Detection**: Extending these two collision detection methods for a system with multiple GPUs would be interesting. We would also like to integrate them to perform collision detection for techniques that adapt the mesh during simulation, such as level-of-detail and multiresolution. Additionally, new GPU features, such as dynamic parallelism, hyper Q and grid management unit, may unlock potential to better parallelise collision detection computation.

- **Further Investigation of Additional Effects of Friction**: In addition to cloth, animating rigid and deformable objects with the addition of velocity-dependent effects would be beneficial. Considering the improved friction model as part of the implicit solver remains an area for future work as well.

Bibliography

J. O. Ajayi. Fabric smoothness, friction, and handle. *Textile Research Journal*, 62(1):52–59, 1992.

D. A. Alcantara, A. Sharf, F. Abbasinejad, S. Sengupta, M. Mitzenmacher, J. D. Owens, and N. Amenta. Real-time parallel hashing on the gpu. *ACM Trans. Graph.*, 28(5):154:1–154:9, Dec. 2009. ISSN 0730-0301. doi: 10.1145/1618452.1618500.

P. Alliez, N. Laurent, H. Sanson, and F. Schmitt. Efficient view-dependent refinement of 3d meshes using 3-subdivision. In *The Visual Computer*, volume 19, pages 205–221, 2003.

F. Altpeter. *Friction Modeling, Identification and Compensation.* PhD thesis, Lausanne, 1999.

R. D. Anandjiwala and G. A. V. Leaf. Large-scale extension and recovery of plain woven fabrics : Part i: Theoretical. *Textile Research Journal*, 61(11):619–634, 1991a.

R. D. Anandjiwala and G. A. V. Leaf. Large-scale extension and recovery of plain woven fabrics : Part ii: Experimental and discussion. *Textile Research Journal*, 61(12):743–755, 1991b.

B. Armstrong-Hélouvry, P. Dupont, and C. Canudas de Wit. A survey of models, analysis tools and compensation methods for the control of machines with friction. *Automatica*, 30 (7):1083–1138, 1994. ISSN 0005-1098.

S. Bandi, S. B, and D. Thalmann. An adaptive spatial subdivision of the object space for fast collision detection of animating rigid bodies. In *Computer Graphics Forum*, volume 14, pages 256–270, 1995.

D. Baraff and A. Witkin. Large steps in cloth simulation. In *Proceedings of the 25th annual*

conference on Computer graphics and interactive techniques, SIGGRAPH '98, pages 43–54, New York, NY, USA, 1998. ACM.

N. Bell and M. Garland. Efficient sparse matrix-vector multiplication on CUDA. NVIDIA Technical Report NVR-2008-004, NVIDIA Corporation, Dec. 2008.

N. Bell and M. Garland. Implementing sparse matrix-vector multiplication on throughput-oriented processors. In *SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, pages 1–11, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-744-8. doi: http://doi.acm.org/10.1145/1654059.1654078.

M. Bergou, M. Wardetzky, D. Harmon, D. Zorin, and E. Grinspun. A quadratic bending model for inextensible surfaces. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, SGP '06, pages 227–230, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

P. Bliman and M. Sorine. Easy-to-use realistic dry friction models for automatic control. In *Proceedings of 3rd European Control Conference*, pages 3788–3794, 1995a.

P.-A. Bliman and M. Sorine. Easy-to-use realistic dry friction models for automatic control. In *the 3rd European Control Conference*, pages 3788–3794, Rome, Italy, 1995b.

J. Bolz, I. Farmer, E. Grinspun, and P. Schröder. Sparse matrix solvers on the gpu: conjugate gradients and multigrid. *ACM Trans. Graph.*, 22(3):917–924, July 2003. ISSN 0730-0301. doi: 10.1145/882262.882364.

D. E. Breen, D. H. House, and P. H. Getto. A physically-based particle model of woven cloth. *The Visual Computer*, 8:264–277, 1992. ISSN 0178-2789.

D. E. Breen, D. H. House, and M. J. Wozny. Predicting the drape of woven cloth using interacting particles. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, SIGGRAPH '94, pages 365–372, New York, NY, USA, 1994. ACM. doi: 10.1145/192161.192259.

R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pages 594–603, New York, NY, USA, 2002. ACM.

R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 28–36, Aire-la-Ville, Switzerland, Switzerland, 2003a. Eurographics Association. ISBN 1-58113-659-5.

R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, pages 28–36. Eurographics Association, 2003b. ISBN 1-58113-659-5.

B. Chapman. The importance of interfiber friction in wrinkling. *Textile Research Journal*, 45(12):825–829, 1975a.

B. Chapman. Determination of the rheological parameters of fabrics in bending. *Textile Research Journal*, 45(2):137–144, 1975b.

Z. Chen, R. Feng, and H. Wang. Modeling friction and air effects between cloth and deformable bodies. *ACM Trans. Graph.*, 32(4):88:1–88:8, July 2013. ISSN 0730-0301. doi: 10.1145/2461912.2461941.

K.-J. Choi and H.-S. Ko. Stable but responsive cloth. *ACM Trans. Graph.*, 21(3):604–611, July 2002. doi: 10.1145/566654.566624.

T. Clapp, H. Peng, T. Ghosh, and J. Eischen. Indirect measurement of the moment-curvature relationship for fabrics. *Textile Research Journal*, 60(9):525–533, 1990.

O. Comas, Z. Taylor, J. Allard, S. Ourselin, S. Cotin, and J. Passenger. Efficient nonlinear fem for soft tissue modelling and its gpu implementation within the open source framework sofa. In F. Bello and P. Edwards, editors, *Biomedical Simulation*, volume 5104 of *Lecture Notes in Computer Science*, pages 28–39. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-70520-8. doi: 10.1007/978-3-540-70521-5_4.

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, pages 61–69, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-983-8.

A. Das, V. K. Kothari, and N. Vandana. A study on frictional characteristics of woven fabrics. *AUTEX Research Journal*, 5(3):133–140, 2005.

G. Debunne, M. Desbrun, M.-P. Cani, and A. H. Barr. Dynamic real-time deformations using space & time adaptive sampling. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 31–36, New York, NY, USA, 2001. ACM. doi: 10.1145/383259.383262.

B. Eberhardt, A. Weber, and W. Strasser. A fast, flexible, particle-system model for cloth draping. *Computer Graphics and Applications, IEEE*, 16(5):52–59, 1996. ISSN 0272-1716. doi: 10.1109/38.536275.

S. A. Ehmann and M. C. Lin. Accurate and fast proximity queries between polyhedra using convex surface decomposition. *Computer Graphics Forum*, 20(3):500–511, 2001. ISSN 1467-8659. doi: 10.1111/1467-8659.00543.

J. W. Eischen and R. Bigliani. Cloth modeling and animation. chapter Continuum versus particle representations, pages 79–122. A. K. Peters, Ltd., Natick, MA, USA, 2000. ISBN 1-56881-090-3.

M. Eitz and G. Lixu. Hierarchical spatial hashing for real-time collision detection. In *Proceedings of the IEEE International Conference on Shape Modeling and Applications 2007*, pages 61–70, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2815-5. doi: 10.1109/SMI.2007.18.

O. Etzmubeta, M. Keckeisen, and W. Strabetaer. A fast finite element solution for cloth modelling. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, PG '03, pages 244–, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-2028-6.

W. Fan, B. Wang, J.-C. Paul, and J. Sun. A hierarchical grid based framework for fast collision detection. *Computer Graphics Forum*, 30(5):1451–1459, 2011.

X. Feng, H. Jin, R. Zheng, K. Hu, J. Zeng, and Z. Shao. Optimization of sparse matrix-vector multiplication with variant csr on gpus. In *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*, pages 165–172, 2011. doi: 10.1109/ICPADS.2011.91.

K. Fetfatsidis, J. Sherwood, J. Chen, and D. Jauffres. Characterization of the fabric/tool and fabric/fabric friction during the thermostamping process. *International Journal of Material Forming*, 2:165–168, 2009. ISSN 1960-6206.

C. Fnfzig and D. W. Fellner. Easy realignment of k-dop bounding volumes, 2003.

A. Garg, E. Grinspun, M. Wardetzky, and D. Zorin. Cubic shells. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '07, pages 91–98, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

S. Gottschalk, M. C. Lin, and D. Manocha. Obbtree: A hierarchical structure for rapid interference detection, 1996.

S. L. Grand. *Gpu gems 3*, chapter Broad-Phase Collision Detection with CUDA. In Nguyen [2007], first edition, 2007. ISBN 9780321545428.

S. Green. Cloth simulation, 2004. HPC Research Group. http://developer.nvidia.com/object/demo_cloth_simulation.html.

E. Grinspun, P. Krysl, and P. Schröder. Charms: a simple framework for adaptive simulation. *ACM Trans. Graph.*, 21(3):281–290, July 2002. ISSN 0730-0301.

E. Grinspun, A. N. Hirani, M. Desbrun, and P. Schröder. Discrete shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, pages 62–67, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association. ISBN 1-58113-659-5.

D. Harmon, E. Vouga, R. Tamstorf, and E. Grinspun. Robust treatment of simultaneous collisions. In *ACM SIGGRAPH 2008 papers*, SIGGRAPH '08, pages 23:1–23:4, New York, NY, USA, 2008. ACM. ISBN 978-1-4503-0112-1.

M. Harris. *Gpu gems 3*, chapter Parallel Prefix Sum (Scan) with CUDA. In Nguyen [2007], first edition, 2007. ISBN 9780321545428.

J. M. Kaldor, D. L. James, and S. Marschner. Simulating knitted cloth at the yarn level. *ACM Trans. Graph.*, 27(3):65:1–65:9, Aug. 2008. ISSN 0730-0301. doi: 10.1145/1360612.1360664.

J. M. Kaldor, D. L. James, and S. Marschner. Efficient yarn-based cloth with adaptive contact linearization. *ACM Trans. Graph.*, 29(4):105:1–105:10, July 2010. ISSN 0730-0301. doi: 10.1145/1778765.1778842.

A. Kalyanaraman. Coefficient of friction between yarns and contact surfaces. *Indian Journal of Textile Research*, 13:1–6, 1988.

D. Karnopp. Computer simulation of stick-slip friction in mechanical dynamic systems. *Journal of Dynamic Systems, Measurement, and Control*, 107:100–103, 1985.

B. Kevelham and N. Magnenat-Thalmann. Fast and accurate gpu-based simulation of virtual garments. In *Proceedings of the 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry*, VRCAI '12, pages 223–226, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1825-9. doi: 10.1145/2407516.2407570.

Khronos. Opencl: Open computing language.

B.-C. Kim, S. Oh, and K. Wohn. Persistent wrinkles and folds of clothes. *The International Journal of Virtual Reality*, 10(1):61–66, 2011.

D. Kim, J.-P. Heo, J. Huh, J. Kim, and S.-E. Yoon. Hpccd: Hybrid parallel continuous collision detection using cpus and gpus. *Comput. Graph. Forum*, 28(7):1791–1800, 2009.

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*, 4:21–36, 1998.

L. Kobbelt. √3-subdivision. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 103–112, 2000. ISBN 1-58113-208-5.

R. R. Kroiss. Collision detection using hierarchical grid spatial partitioning on the gpu, 2013.

J. Krüger and R. Westermann. Linear algebra operators for gpu implementation of numerical algorithms. *ACM Trans. Graph.*, 22(3):908–916, July 2003. ISSN 0730-0301. doi: 10.1145/882262.882363.

T. J. Lahey. Modelling hysteresis in the bending of fabrics. Master's thesis, University of Waterloo, 2002.

E. S. Larsen and D. McAllister. Fast matrix multiplies using graphics hardware. In *Proceedings of the 2001 ACM/IEEE conference on Supercomputing (CDROM)*, Supercomputing '01, pages 55–55, New York, NY, USA, 2001. ACM. ISBN 1-58113-293-X. doi: 10.1145/582034.582089.

T. Larsson and T. Akenine-Möller. A dynamic bounding volume hierarchy for generalized collision detection. *Comput. Graph.*, 30(3):450–459, June 2006. ISSN 0097-8493. doi: 10.1016/j.cag.2006.02.011.

C. Lauterbach, M. Garland, S. Sengupta, D. Luebke, and D. Manocha. Fast bvh construction on gpus. *Computer Graphics Forum*, 28(2):375–384, 2009.

C. Lauterbach, Q. Mo, and D. Manocha. gproximity: Hierarchical gpu-based operations for collision and distance queries. *Computer Graphics Forum*, pages 419–428, 2010.

H. Li, Y. Wan, and G. Ma. A cpu-gpu hybrid computing framework for real-time clothing animation. In *Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on*, pages 391–396, 2011. doi: 10.1109/CCIS.2011.6045096.

F. Liu, T. Harada, Y. Lee, and Y. J. Kim. Real-time collision culling of a million bodies on graphics processing units. *ACM Trans. Graph.*, 29:154:1–154:8, December 2010. ISSN 0730-0301.

B. Lloyd, G. Szekely, and M. Harders. Identification of spring parameters for deformable object simulation. *Visualization and Computer Graphics, IEEE Transactions on*, 13(5): 1081–1094, 2007. ISSN 1077-2626. doi: 10.1109/TVCG.2007.1055.

C. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, The University of Utah, 1987.

J. Mezger, S. Kimmerle, and O. Etzmu. Hierarchical techniques in collision detection for cloth animation. *Journal of WSCG*, 11(2):322–329, 2003.

B. Mirtich. Hybrid simulation: Combining constraints and impulses, 1996.

B. Mirtich. Efficient algorithms for two-phase collision detection. Technical report, 1997.

B. Mirtich and J. F. Canny. Impulse-based simulation of rigid bodies. In *SI3D*, pages 181–188, 1995.

M. Moore and J. Wilhelms. Collision detection and response for computer animation. *SIGGRAPH Comput. Graph.*, 22(4):289–298, June 1988. ISSN 0097-8930. doi: 10.1145/378456.378528.

A. Morin. New friction experiments carried out at metz in 18311833. *Proceedings of the French Royal Academy of Sciences*, 4:1–128, 1833.

M. Müller. Hierarchical position based dynamics. In *VRIPHYS*, pages 1–10, 2008.

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

M. Müller, J. Dorsey, L. McMillan, R. Jagnow, and B. Cutler. Stable real-time deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '02, pages 49–54, New York, NY, USA, 2002. ACM. ISBN 1-58113-573-4. doi: 10.1145/545261.545269.

M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position based dynamics. *J. Vis. Comun. Image Represent.*, 18(2):109–118, Apr. 2007. ISSN 1047-3203. doi: 10.1016/j.jvcir.2007.01.005.

C. Ngoc and S. Boivin. Nonlinear Cloth Simulation. (RR-5099), 2004.

H. Nguyen. *Gpu gems 3*. Addison-Wesley Professional, first edition, 2007. ISBN 9780321545428.

NVIDIA. Cuda: Compute unified device architecture.

J. F. O'Brien, A. W. Bargteil, and J. K. Hodgins. Graphical modeling and animation of ductile fracture. *ACM Trans. Graph.*, 21(3):291–294, July 2002. ISSN 0730-0301. doi: 10.1145/566654.566579.

H. Olsson, K. J. Astrom, C. C. de Wit, M. Gafvert, and P. Lischinsky. *Friction models and friction compensation*. 1998.

M. A. Otaduy, O. Chassot, D. Steinemann, and M. Gross. Balanced hierarchies for collision detection between fracturing objects. *2013 IEEE Virtual Reality (VR)*, 0:83–90, 2007. doi: http://doi.ieeecomputersociety.org/10.1109/VR.2007.352467.

J. Owen. The bending behaviour of plain-weave fabrics woven from spun yarns. *Journal of The Textile Institute*, 59:313–343, 1968.

S. Pabst, S. Krzywinski, A. Schenk, and B. Thomaszewski. Seams and bending in cloth simulation. In F. Faure and M. Teschner, editors, *VRIPHYS*, pages 31–38. Eurographics Association, 2008. ISBN 978-3-905673-70-8.

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

S. Pabst, A. Koch, and W. Straer. Fast and scalable cpu/gpu collision detection for rigid and deformable surfaces. *Computer Graphics Forum*, 29(5):1605–1612, 2010. ISSN 1467-8659.

I. J. Palmer and R. L. Grimsdale. Collision detection for animation using sphere-trees. *Computer Graphics Forum*, 14(2):105–116, 1995. ISSN 1467-8659.

X. Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *In Graphics Interface*, pages 147–154, 1996.

X. Provot. *Collision and self-collision handling in cloth model dedicated to design garments*, volume 97, pages 177–189. Citeseer, 1997.

A. Rasmusson, J. Mosegaard, and T. S. Sørensen. Exploring parallel algorithms for volumetric mass-spring-damper models in cuda. In *Proceedings of the 4th international symposium on Biomedical Simulation*, ISBMS '08, pages 49–58, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-70520-8. doi: 10.1007/978-3-540-70521-5_6.

O. Reynolds. On the theory of lubrication and its application to mr. beauchamp tower's experiments, including an experimental determination of the viscosity of olive oil. *Philosophical Transactions of the Royal Society of London*, 177:157–234, 1886.

J. Rodrïguez-Navarro and A. Susïn. Non structured meshes for cloth gpu simulation using fem. In *3rd. Workshop in Virtual Reality, Interactions, and Physical Simulations*, pages 1–7, 2006.

J. Rodrïguez-Navarro, M. Sainz, and A. Susïn. Gpu based cloth simulation with moving humanoids. In *Actas XV Congreso Espaol de Informtica Grfica*, pages 147–155, 2005. ISBN 84-9732-431-5.

M. Rumpf and R. Strzodka. Nonlinear diffusion in graphics hardware. In *Proceedings of the 3rd Joint Eurographics - IEEE TCVG conference on Visualization*, EGVISSYM'01, pages

75–84, Aire-la-Ville, Switzerland, Switzerland, 2001a. Eurographics Association. ISBN 3-211-83674-8. doi: 10.2312/VisSym/VisSym01/075-084.

M. Rumpf and R. Strzodka. Using graphics cards for quantized fem computations. In *VIIP*, pages 193–202, 2001b.

F. Schornbaum. Hierarchical hash grids for coarse collision detection. Master's thesis, Studienarbeit, 2009.

C. A. Schroeder, N. Kwatra, W. Zheng, and R. Fedkiw. Asynchronous evolution for fully-implicit and semi-implicit time integration. *Comput. Graph. Forum*, pages 1983–1992, 2011.

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 Computer Graphics*, 15(2):339–350, 2009.

S. Sengupta, M. Harris, Y. Zhang, and J. D. Owens. Scan primitives for gpu computing. In *Graphics Hardware 2007*, pages 97–106. ACM, Aug. 2007.

J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. 1994.

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, Mar. 1989. ISSN 0045-7825.

R. Stribeck. The key qualities of sliding and roller bearings. *Zeitschrift des Vereines Seutscher Ingenieure*, 46:1342–1348, 1902.

M. Tang, S. Curtis, S.-E. Yoon, and D. Manocha. Interactive continuous collision detection between deformable models using connectivity-based culling. In *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, SPM '08, pages 25–36, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-106-4. doi: http://doi.acm.org/10.1145/1364901.1364908. URL http://doi.acm.org/10.1145/1364901.1364908.

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*, pages 355–360, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-711-0.

M. Tang, D. Manocha, and R. Tong. Fast continuous collision detection using deforming non-penetration filters. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, I3D '10, pages 7–13, New York, NY, USA, 2010a. ACM.

M. Tang, D. Manocha, and R. Tong. Mccd: Multi-core collision detection between deformable models using front-based decomposition. *Graphical Models*, 72(2):7–23, 2010b. ISSN 1524-0703.

M. Tang, D. Manocha, J. Lin, and R. Tong. Collision-streams: fast gpu-based collision detection for deformable models. In *Symposium on Interactive 3D Graphics and Games*, I3D '11, pages 63–70. ACM, 2011.

M. Tang, D. Manocha, M. A. Otaduy, and R. Tong. Continuous penalty forces. *ACM Trans. Graph.*, 31(4):107:1–107:9, July 2012. ISSN 0730-0301. doi: 10.1145/2185520.2185603.

M. Tang, D. Manocha, R. Narain, R. Tong, and C. Meng. A gpu-based streaming algorithm for high-resolution cloth simulation. In *Tech Report, Univ. of North Carolina at Chapel Hill*, 2013.

P. Taylor and D. Pollet. A novel technique to measure stick-slip in fabric. *International Journal of Clothing Science and Technology*, 12:124–133, 2000.

E. Tejada and T. Ertl. Large steps in gpu-based deformable bodies simulation. *Simulation Modelling Practice and Theory*, 13(8):703715, 2005. ISSN 1569-190X.

D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: viscolelasticity, plasticity, fracture. *SIGGRAPH Comput. Graph.*, 22(4):269–278, June 1988. ISSN 0097-8930.

D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *SIGGRAPH Comput. Graph.*, 21(4):205–214, 1987. ISSN 0097-8930.

M. Teschner, B. Heidelberger, M. Mueller, D. Pomeranets, and M. Gross. Optimized spatial hashing for collision detection of deformable objects. In *Vision Modeling Visualization*, pages 47–54, 2003.

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. *Eurographics State-of-the-Art Report (EG-STAR)*, pages 119–139, 2004.

T. A.-M. Thomas Larsson. Collision detection for continuously deforming bodies. In *European Association for Computer Graphics*, pages 325 – 333, 2001.

B. Thomaszewski, S. Pabst, and W. Strasser. Asynchronous cloth simulation. In *Proceedings of Computer Graphics International*, pages 246–255, 2008.

C. Thompson, S. Hahn, and M. Oskin. Using modern graphics architectures for general-purpose computing: a framework and analysis. In *Microarchitecture, 2002. (MICRO-35). Proceedings. 35th Annual IEEE/ACM International Symposium on*, pages 306–317, 2002. doi: 10.1109/MICRO.2002.1176259.

G. van den Bergen, G. Van, and D. Bergen. Efficient collision detection of complex deformable models using aabb trees. *J. Graphics Tools*, 2, 1998.

A. Van Gelder. Approximate simulation of elastic membranes by triangulated spring meshes. *J. Graph. Tools*, 3(2):21–42, Feb. 1998. ISSN 1086-7651. doi: 10.1080/10867651.1998.10487490.

T. I. Vassilev and B. Spanlang. Fast gpu garment simulation and collision detection. In *Winter School of Computer Graphics*, 2012.

L. Velho and D. Zorin. 4-8 subdivision. In *Computer Aided Geometric Design*, volume 18, pages 397–427, 2001.

P. Volino and N. Magnenat-Thalmann. Implicit midpoint integration and adaptive damping for efficient cloth simulation. *Computer Animation and Virtual Worlds*, 16(3-4):163–175, 2005a. ISSN 1546-427X. doi: 10.1002/cav.78. URL http://dx.doi.org/10.1002/cav.78.

P. Volino and N. Magnenat-Thalmann. Accurate garment prototyping and simulation. *Computer-Aided Design and Applications*, 2(5):645–654, 2005b. doi: 10.1080/16864360.2005.10738329.

P. Volino, N. Magnenat-Thalmann, and F. Faure. A simple approach to nonlinear tensile stiffness for accurate cloth simulation. *ACM Trans. Graph.*, 28(4):105:1–105:16, 2009. ISSN 0730-0301.

Z. Wang, X. Xu, W. Zhao, Y. Zhang, and S. He. Optimizing sparse matrix-vector multiplication on cuda. In *Education Technology and Computer (ICETC), 2010 2nd International Conference on*, volume 4, pages V4–109–V4–113, 2010. doi: 10.1109/ICETC.2010.5529724.

J. Weil. The synthesis of cloth objects. *SIGGRAPH Comput. Graph.*, 20(4):49–54, Aug. 1986. doi: 10.1145/15886.15891. URL http://doi.acm.org/10.1145/15886.15891.

T. Wong, G. Leach, and F. Zambetta. Virtual subdivision for gpu based collision detection of deformable objects using a uniform grid. *The Visual Computer*, 28:829–838, 2012. ISSN 0178-2789.

T. H. Wong, G. Leach, and F. Zambetta. An improved friction model for cloth simulation. In *Pacific Graphics Short Papers*, pages 41–46. The Eurographics Association, 2013a.

T. H. Wong, G. Leach, and F. Zambetta. Modelling bending behaviour in cloth simulation using hysteresis. *Computer Graphics Forum*, 32(8):183–194, 2013b. ISSN 1467-8659. doi: 10.1111/cgf.12196.

T. H. Wong, G. Leach, and F. Zambetta. An adaptive octree grid for gpu-based collision detection of deformable objects. Manuscript submitted for publication, 2014.

W. S.-K. Wong and G. Baciu. A randomized marking scheme for continuous collision detection in simulation of deformable surfaces. In *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, VRCIA '06, pages 181–188, New York, NY, USA, 2006. ACM. ISBN 1-59593-324-7.

P. Wriggers, T. V. Van, and E. Stein. Finite element formulation of large deformation impact-contact problems with friction. *Computers & Structures*, 37(3):319 – 331, 1990. ISSN 0045-7949. doi: http://dx.doi.org/10.1016/0045-7949(90)90324-U.

X. Wu, M. S. Downes, T. Goktekin, and F. Tendick. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. *Computer Graphics Forum*, 20(3):349–358, 2001. ISSN 1467-8659. doi: 10.1111/1467-8659.00527.

C. Yuksel, J. M. Kaldor, D. L. James, and S. Marschner. Stitch meshes for modeling knitted clothing with yarn-level detail. *ACM Trans. Graph.*, 31(4):37:1–37:12, July 2012. ISSN 0730-0301. doi: 10.1145/2185520.2185533.

C. Zeller. Cloth simulation on the gpu. In *ACM SIGGRAPH 2005 Sketches*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

D. Zhang and M. M. F. Yuen. Collision detection for clothed human animation. In *Proceedings of the 8th Pacific Conference on Computer Graphics and Applications*, PG '00, pages 328–, Washington, DC, USA, 2000. IEEE Computer Society. ISBN 0-7695-0868-5.

L. Zhao, L. Qin, F. Wang, and H. H. H. Chuah. Factors affecting recovery of ptt shape memory fabric to its initial shape. *International Journal of Clothing Science and Technology*, 21(1):64–73, 2009.