25th Annual Conference of the European Association for Computer Graphics
“Interacting with Virtual Worlds”
Grenoble, August 30th – September 3rd

Tutorial 1

Simulation of Clothes for Real-time Applications

N. Magnenat-Thalmann, F. Cordier, P. Volino
MIRALab, University of Geneva
M. Keckeisen, S. Kimmerle
University of Tübingen
R. Klein, J. Meseth
University of Bonn

Published by INRIA and the Eurographics Association, ISSN 1017-4656

INRIA
RHÔNE-ALPES

RHÔNE-ALPES
LA RÉGION

isère
Conseil Général
Plus proche de vous !

LA METRO.

VILLE DE GRENOBLE

IMAG

INP Grenoble

GRENOBLE 1
UNIVERSITE
JOSEPH FOURIER
SCIENCES, TECHNOLOGIE, MEDECINE

ATi™

IBM Research

nVIDIA®

APOM
ASSOCIATION DES PRODUCTEURS D'OEUVRES MULTIMEDIA

Grenoble Interactive

LyonGame

CORNER HOUSE

rhône-alpes
numérique

Simulation of Clothes for Real-time Applications

Organized by: N. Magnenat-Thalmann
Presenters: N. Magnenat-Thalmann, F. Cordier, P. Volino
MIRALab, University of Geneva
M. Keckeisen, S. Kimmerle
University of Tübingen
R. Klein, J. Meseth
University of Bonn

Full day tutorial
Time: Monday 30th, 8h30-12h / 14h-17h30
Room: Amphi GEM

Keywords

Real-time graphics, real-time cloth deformation, collision detection, virtual heritage, numerical solvers, real-time rendering of textiles, data-driven simulator.

Overview

8:30 – 8:40: Presentation of the Course (N. Magnenat-Thalmann)

8:40 – 9:00: Introduction and State of the Art in Clothing (Univ. of Geneva, p. 7)

This presentation gives an overview of the whole tutorial. It introduces the different techniques that will be presented in the next sessions. An overview of previous work is given as well.

- Overview of the tutorial
- Problematic of simulating clothes in real-time
- Motivating applications
- Overview of the real-time cloth simulation techniques:
    - Physics-based simulation
    - Collision detection
    - Example-based approaches

9:00 – 10:00: Physical Models & Numerical Solvers (Univ. of Tübingen, p. 17)

Physical models and numerical solvers are the basis of current cloth simulation engines. We will discuss physical models for cloth ranging from simple mass-spring systems to continuum-based particle systems and fast finite element solutions. Then, we will describe several explicit and implicit time integration schemes and compare them with respect to stability and performance.

- Physical models
    - Discrete models
    - Continuous models
- Numerical simulation
    - Methods for numerical integration
    - Solving nonlinear systems
    - Comparison of methods

10:00 – 10:30: Break

10:30 – 12:00: Collision Detection (Univ. of Tübingen & Univ. of Geneva, p. 35)

In the simulation of cloth, collision detection is crucial for the quality of the results and for the performance as the collision detection process can be highly complex. We will first discuss the specificity of collision detection for clothes and will thereafter review some recently used techniques for the real-time collision detection of clothes. In the last section we will detail a fast collision detection approach based on bounding volume hierarchies as it is specially suited both for animated as for static scenes and is also able to detect self-collisions.

- Techniques for collision detection on deformable models
    - Distance fields
    - Image-space techniques
- Bounding volumes hierarchies

    - Hierarchies and spatial subdivision
    - Hierarchy traversal and self-collision heuristics
- Detection strategies for deformable objects
- General collision detection on polygonal meshes
- Self-collision detection
- Data-driven collision detection

12:00 – 14:00: Lunch

14:00 – 15:00: Real-time Rendering of Textiles (Univ. of Bonn, p. 51)

Reproducing the complex reflectance properties of complex materials like fabrics at real-time frame rates is an important part of cloth visualization systems. In this part, we will present methods for acquisition, compression and real-time rendering of material reflectance represented as bidirectional texture functions (BTFs). The presented rendering approaches include large-scale shadows and image-based lighting leading to a highly realistic appearance of visualized clothes.

- BTF Acquisition and Compression
    - Automatic HDR BTF acquisition
    - Compression techniques
- High-quality and real-time BTF rendering
    - Hardware-accelerated
    - Image-based lighting
    - BTF ray-tracer

15:00 – 15:30: Context-Specific Cloth Simulation (Univ. of Geneva, p. 69)

Most of the existing approaches use a general-purpose simulation method using collision detection and physical simulation for the whole garment. Unfortunately, simulations that simply calculate all potentially colliding vertices may generate a highly realistic movement, but do not provide a guaranteed frame time. This talk will demonstrate how the computation cost can be greatly reduced by making use of predetermined conditions between the cloth and the body model, avoiding complex collision detection and physical deformations wherever possible.

- Simulation of the cloth dynamics with geometric and physics-based models
- Geometric deformation of cloth wrinkles
- Data-driven clothes simulation

15:30 – 16:00: Break

16:00 – 17:30: Case-studies in National and European Research Projects (Univ. of Bonn & Univ. of Geneva, p. 83)

This session will show several applications of cloth simulation in the frame of European and National Research projects:

- Life Plus
- Cahrisma
- E-Tailor

- RealReflect
- Virtual Try-On

Intended audience:

This tutorial assumes basic background in Computer Graphics. Specific knowledge on physics-based simulation is not necessary.

Presenters' background:

Nadia Magnenat-Thalmann (thalmann@miralab.unige.ch) has contributed to pioneer research into Virtual Humans for over 25 years, and has participated to spectacular state-of-the-art demonstrations and rigorous and intensive academic research programs that make them possible. After her PhD in Quantum Physics and Computer Graphics from the University of Geneva, she has been a Professor at the University of Montreal in Canada from l977 to 1988. In l989, she founded MIRALab, an interdisciplinary Research Lab at the University of Geneva in Switzerland. She has been part of the European project E-Tailor and now Leapfrog.

Frederic Cordier (cordier@miralab.unige.ch) received his PhD in Computer Science from MIRALab, University of Geneva in January 2004. His current research interests include real-time animation of virtual humans, particularly on real-time clothes, skin deformation and interaction among clothes and skin. In this area, he has published several journal papers in Computer Graphics Forum, IEEE Computer Graphics & Applications, Journal on Visualization and Computer Animation, etc.

Pascal Volino (pascal@miralab.unige.ch) is a computer scientist, working at MIRAlab, University of Geneva. He is actually working on new models for cloth animation, involving versatile models for efficient simulations on situations involving high deformation, wrinkling and multilayer garments. The research is particularly focused on data structure, efficient collision detection, robust simulation and interactive cloth manipulation.

Michael Keckeisen (keckeise@gris.uni-tuebingen.de) studied Mathematics and Computer Science at the University of Tübingen and received a Diploma in Mathematics (2000). Since 2001 he is a member of the graphics research group at the University of Tübingen. His main research interests are the physically based modelling and simulation of clothes, and interaction techniques in virtual environments.

Stefan Kimmerle (kimmerle@gris.uni-tuebingen.de) studied physics and chemistry in Tübingen and San Diego. In 2000 he received his Diploma in physics from the University of Tübingen. Since 2001 he is a Ph.D. student at the graphics research group at GRIS. In 2003 he was an invited researcher at GRAVIR, INRIA Rhône-Alpes in Grenoble. His main research interests are physically-based modelling and collision detection for deformable objects.

Reinhard Klein (rk@cs.uni-bonn.de) received his Ph.D. in computer science from the University of Tübingen, Germany, in 1995. After receiving an appointment as lecturer ("Habilitation") in computer science in 1999, he became an Associate Professor at the

University of Darmstadt, Germany. Since October 2000, Reinhard Klein is full professor at the University of Bonn and head of the department of Computer Science II.

Jan Meseth (meseth@cs.uni-bonn.de) is a research assistant and Ph.D. student in the Computer Graphics Group at the University of Bonn, Germany. His main research interests are photo-realistic, image-based rendering and level-of detail representations.

State of the Art in Clothing

(Tutorial Notes)

Pascal Volino, Frederic Cordier, Nadia Magnenat-Thalmann

MIRALab, CUI, University of Geneva, Switzerland

E-mail: {volino, cordier, thalmann}@miralab.unige.ch

Abstract

Virtual garment design and simulation involves a combination of a large range of techniques, involving mechanical simulation, collision detection, and user interface techniques for creating garments. Here, we perform an extensive review of the evolution of these techniques made in the last decade to bring virtual garments to the reach of computer applications not only aimed at graphics, but also at CAD techniques for the garment industry.

1 Introduction

The challenges of virtual garment simulation are numerous, and have attracted research efforts for more than a decade. First dedicated to the realistic simulation of the mechanical behavior of cloth, it soon evolved towards simulation of virtual garments on synthetic characters. While computer graphics gets the most obvious benefits from garment simulation on animated virtual characters, virtual prototyping of garment models is another major application field for the garment industry.

Virtual garment simulation is the result of a large combination of techniques that have also dramatically evolved during the last decade. Unlike the mechanical models used for existing mechanical engineering for simulating deformable structures, a lot of new challenges arise from the highly versatile nature of cloth. The central pillar of garment simulation obviously remains the development of efficient mechanical simulation models, which can accurately reproduce with the specific mechanical properties of cloth. However, cloth is by nature highly deformable and specific simulation problems arise from this fact. First, the mechanical representation should be accurate enough to deal with the nonlinearities and large deformations occurring at any place in the cloth, such as folds and wrinkles. Moreover, the garment cloth interacts strongly with the body that wears it, as well as with the other garments of the apparel. This requires advanced methods for efficiently detecting the geometrical contacts constraining the behavior of the cloth, and to integrate them in the mechanical model (collision detection and response). All these methods require advanced and complex computational methods where most important key issues remain computation speed and efficiency. For real-time applications however, only specific approximation and simplification methods allow the computation of garment animation, giving up some of the mechanical accuracy of the result in a result rather focused on visual realism.

Garment simulation, which started in the late eighties with very simple models such as Weil's approach [WEI 86], has taken much benefit from the increasing performance of computer hardware and tools as well as the development of specific simulation technologies which have nowadays lead to impressive applications not only in the field of simulation of virtual worlds, but also as design tools for the garment and fashion industry.

This chapter reviews the developments of these techniques, from early beginnings (§ 2) to the state-of-the-art methods in cloth simulation and collision processing (§ 3,4,5), leading to the existing current developments in garments.

2 The Beginnings of Garment Simulation

In the field of computer graphics, the first applications for mechanical cloth simulation appeared in 1987 with the work of Terzopoulos et al [TER 87] [TER 88] in the form of a simulation system relying on the Lagrange equations of motion and elastic surface energy. Solutions were obtained through finite difference schemes on regular grids. This allowed simple scenes involving cloth to be simulated, such as the accurate simulation of a flag or the draping of a rectangular cloth. However, the first applications that really simulated garments started in 1990 (Figure 1) with the considerations of many other technologies complementing cloth simulation [LAF 91] [CAR 92], such as body modelling and animation, and collision detection and response [YAN 93]. These applications innovated by providing the first virtual system allowing virtual garment patterns to be sewed together around a character.

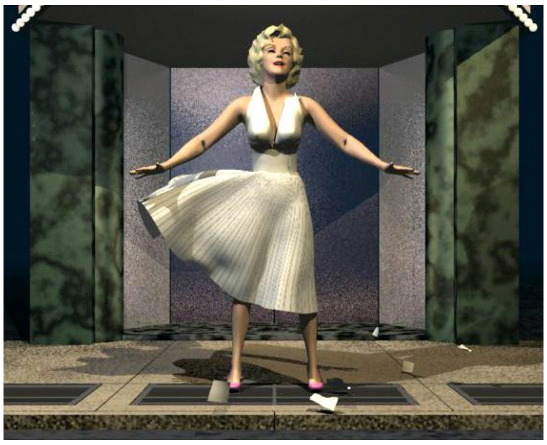


Figure 1: *"FlashBack": Early virtual garments used context-dependent simulation of simplified cloth models.*

Since then, most developments were aimed at optimizing the accuracy and efficiency of the methods for simulating cloth accurately and efficiently, along the developments of actual applications and commercial products.

3 Mechanical Models

The accurate reproduction of the mechanical behaviour of cloth has always been a key issue for garment simulation. The mechanical behaviour of cloth is usually measured using standardized protocols, such as the Kawabata Evaluation System (KES), or the simpler FAST method, which are based on the experimental measurement of strain-stress curves for elongation, shearing and bending on normalized samples of fabric. Different representations of the cloth surface mechanics then allow the virtual reproduction of the behaviour of cloth.

Well known in mechanical engineering, the Finite Element method considers the cloth surface as being discretized in interpolation patches for a given order (bilinear, trilinear, quadrilinear), and an associated set of parameters (degrees of freedom) that give the actual shape to the interpolation surface over the element. From the mechanical properties of the material, the mechanical energy is computed from the deformation of the surface for given values of the interpolation parameters. An equation system based on the energy variation is then constructed with these degrees of freedom. Surface continuity between adjacent elements imposes additional constraint relationships. A large sparse linear system is built by assembling successively the contributions of all the elements of the surface, and then solved using optimized iterative techniques, such as the conjugate gradient method.

Finite elements have only had a marginal role in cloth simulation. The main attempts are described in [COL 91], [GAN 95], [EIS 96]. Most implementations focus on the accurate reproduction of mechanical properties of fabrics, but restrict the application field to the simulation of simple garment samples under elementary mechanical contexts, mostly because of the huge computational requirements of these models. Furthermore, accurate modelling of highly variable constraints (large nonlinear deformations, highly variable collisions) is difficult to integrate into the formalism of finite elements, and this sharply reduces the ability of the model to cope with the very complicated geometrical contexts which can arise in real-world garment simulation on virtual characters.

An easier and more pragmatic way to perform cloth simulation is the use of particle systems. Particle systems consider the cloth to be represented only by the set of vertices that constitute the polygonal mesh of the surface. These particles are moved through the action of forces that represent the mechanical behaviour of the cloth, which are computed from the geometric relationships between the particles that measure the deformation of the virtual cloth. Among the different variations of particle systems, the spring-mass scheme is the simplest and most widely used (Figure 2). It considers the distance between neighbouring particle pairs as the only deformation measurement and interaction source representing the internal elasticity of the cloth.

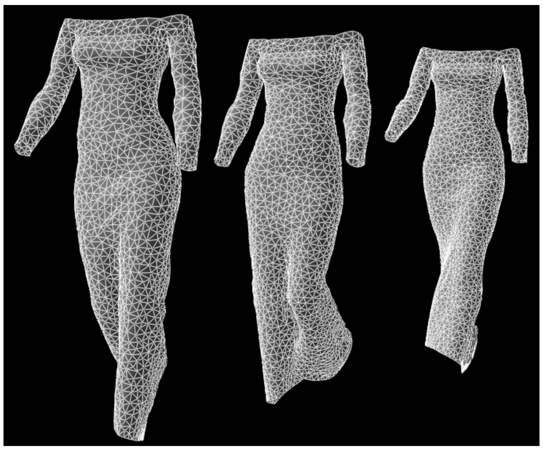


Figure 2: *Particle systems model cloth surfaces as point masses interacting with their neighbours using forces computed from their relative positions.*

Particle systems are among the simplest and most efficient ways to define rough models that compute highly deformable mechanical systems such as cloth with computation times small enough to integrate them into systems for simulating complete garments on virtual bodies. Among the main contributions on particle system models, early works considered simple viscoelastic models on regular grids with applications for draping problems with simple numerical integration schemes [SAK 91]. Accurate models started with Breen et al [BRE 94] on modelling the microstructure of cloth using parameters derived from KES behaviour curves and integration based on energy minimization. However, such accurate models required a lot of computation for solving problems that were restricted to draping. On the other hand, more recent models trade accuracy for speed, such as the grid model detailed by Provot et al [PRO 95] which additionally includes geometric constraints for limiting large deformation of cloth. Additional contributions from Eberhardt et al [EBE 96] with the simulation of KES parameters and comparison of the efficiency of several integration methods. Advanced surface representations were used in [DER 98], where the simulation model and collision detection takes advantage of the hierarchical structure of subdivision surfaces. Modelling animated garments on virtual characters is the specific aim of the work described by Volino et al [VOL 95] [VOL 97], which investigate improved spring-mass representations for better accuracy of surface elasticity modelling on irregular meshes.

While various models can be used to compute the force applied on each particle given their position and speed, these forces have then to be integrated along time to obtain the position and speed of the particle for the following time-steps using methods related to the integration of ordinary differential equation systems. Most recent however focus on improvements of the numerical integration methods in order to improve efficiency of the simulation.

Explicit integration methods are the simplest methods available for solving first-order ordinary differential systems. They consider the prediction of the future system state directly from the value of the derivatives. The best known techniques are the Runge-Kutta methods. Among them, the fast but unstable and inaccurate first-order Euler method, used in many early implementations, considers the future state as a direct extrapolation from the current state and the derivative. Higher order and more accurate methods also exist, such as the second-order Midpoint method, used for instance in early models by Volino et al [VOL 95], and the very accurate fourth-order Runge-Kutta method, used for instance by Eberhardt et al [EBE 96].

Beside considerations for accuracy, stability and robustness are other key factors to consider. For most situations encountered in cloth simulation, the numerical stiffness of the equations (stiff elastic forces, small surface elements) require the simulation time-steps to be small enough to ensure the stability of the system, and this limits the computation speed much more than accuracy considerations. Adequate time-step control is therefore essential for an optimal simulation. A common solution is to use the fifth-order Runge-Kutta algorithm detailed in [PRE 92] which embeds integration error evaluation used for tuning the time-step adaptively [VOL 97].

In order to circumvent the problem of instability, implicit numerical methods are being used. For cloth simulation, this was first outlined by Baraff et al [BAR 98]. The most basic implementation of implicit method is the Euler step, which considers finding the future state for which "backward" Euler computation would return the initial state. It performs the computation not using the derivative at the current time-step, but using the predicted derivative for the next time-step. Besides the inverse Euler method, other, more accurate higher-order implicit methods exist, such as the inverse Midpoint method, which remains quite simple but exhibits some instability problems. A simple solution is to interpolate between the equations of the Euler and Midpoint methods, as proposed by Volino et al [VOL 00]. Higher-order methods, such as the Rosenbrook method, however do not exhibit convincing efficiencies in the field of cloth simulation. Multi-step methods, which perform a single-step iteration using a linear combination of several previous states, are other good candidates for a good accuracy-stability compromise. Among them, the second-order Backward Differential Formula (BDF-2) has shown some interesting performances, as used by Eberhardt, Hauth et al [EBE 00] [HAU 01] and Choi et al [CHO 02].

Whatever variation chosen, the major difficulty in using implicit integration methods is that they involve the resolution of a large and sparse linear equation system for each iteration, constructed from the Jacobian matrix of the particle forces against their position and speed. A commonly used simplification involves linearization of the mechanical model so as to obtain a linear approximation of the matrix that does not evolve along time, and on which initial construction and pre-processing allows efficient resolution method to be used, as for example like Kang et al [KAN 00], or even the matrix inverse to be pre-computed as done by Desbrun et al [DES 99]. A further simplification is to suppress completely the need of computing the matrix using an adapted approximation embedded directly in an explicit iteration. A big drawback of all these methods results from the approximation of the matrix that cannot take into account the nonlinearities of the model (mostly those resulting from the change of orientation of the surface elements during the simulation). While this is acceptable for draping applications, animations behave usually poorly because of excessive numerical damping, which also increases as the time-step becomes large.

The best numerical method for actually resolving the linear system seems to be the Conjugate Gradient method, as suggested by Baraff et al [BAR 98], with various variations and preconditioning schemes depending on how the mechanical model is formulated and geometrical constraints of the cloth integrated.

Most models using implicit integration schemes restrict themselves to using spring-mass systems, as their simple formulation eases the process of defining the linear system to be resolved. However, implicit integration methods can also be used for integrating accurate surface-based particle systems as the one described above, from derivation of the particle force expressions relatively to the particle positions and speeds. This in quite simply integrated into the implicit formulations described by Volino et al [VOL 00], and extended toward other advanced methods as by Hauth et al [HAU 01]. These formulations actually blur the line between particle systems and finite element methods, as the described particle system is indeed a first-order finite element method where the implicit resolution scheme corresponds to the energy minimization scheme of finite elements and the build of the linear system matrix to the assembly process of elements into the global system to be resolved. This is a key idea to design a new system which combines the accuracy of finite elements with the efficiency of the techniques used for particle systems, as described in section 3.

4 Real-Time Garment Animation

Real-time garment animation poses the challenging problem of how to perform very fast methods for the mechanical computation and collision detection. Accuracy has to be given up in favour of quicker methods that take advantage of geometrical approximations and contextual simplifications.

4.1 Physically-based simulation

Implicit integration [BAR 98] [CHO 02] and speed-optimized derivatives [DES 99] [MEY 00] allow fast simulation of mechanical properties of cloth. However, the computation speed still remains slow for complex garments, and these methods are still limited by the maximum number of polygons they can animate in real-time.

In the specific of area optimizations for mechanical behaviour, James et al. [JAM 99] have worked on real time simulation. Their paper describes the boundary integral equation formulation of static linear elasticity as well as the related Boundary Element Method discretization technique. Their model is not dynamic, but rather a collection of static postures, limiting its potential applications. Debunne et al. [DEB 00] have also recently introduced a technique for animating soft bodies in real time. However, their method works on volumetric meshes and is therefore not applicable to thin objects such as cloth.

4.2 Collision detection

Collision detection is another bottleneck in the speed of cloth simulation. Besides the traditional methods, specific optimizations intend to address the problem of real-time simulation. For instance, Vassilev et al. [VAS

01] propose to use z-buffer for collision detection in order to generate depth and normal maps. The computation time of their collision detection does not depend on the complexity of the body. However, the maps need to be pre-computed before simulation, also restricting the real-time application. This class of methods has become very popular recently [GOV 03] [BAC 02]; high performance 3D graphic systems are part of almost every personal computer or game console.

Another approach "voxel-based collision" has also been experimented. This approach is based on a spatial subdivision. The algorithm starts by selecting the voxel size that meets the requirements of targeted accuracy and performance. It then partitions the space into regions of free space, object surface and object interior. All voxels are organized hierarchically to form an octree. Voxel-based collision detection has been implemented by Meyer et al. [MEY 00] and McNeely et al. [MCN 99]. The collision detection remains as computing which voxel the cloth vertex belongs to and checking the flags of the voxel to know whether the cloth vertex is in collision or not. The main advantage of this algorithm is its rapid computation speed. On the other hand, the space partitioning needs to be calculated during the pre-processing, restricting the collision detection to rigid objects only.

4.3 Geometric approaches

Some other researchers have used geometrical approaches [WEI 86] [AGU 90] [HIN 90] [NGG 95]. Geometrical models do not consider the physical properties of the cloth, therefore providing techniques that produce fast results. However, these techniques are not able to reproduce the dynamics of clothes. Moreover, geometrical techniques require a considerable degree of user intervention. They can be regarded as a form of advanced drawing tools.

4.4 Hybrid approaches

Hybrid approaches aim to combine nicely physically based deformation and geometric deformation. The idea of hybrid approach stems from the following observation: physically based simulations are slow to compute but produce realistic results while simulations based on geometric method are much faster but not really suitable to animate full clothes. By combining advantages of the both approaches, one can expect to have acceptable results within moderate computation time. In most cases, physically based models are used to compute the global movements of garments and the details such as wrinkles are generated with geometric models.

Kang et al. [KAN 01] [KAN 02] improved the visual quality of the garments of small number of polygons by tessellating the triangles. With a cubic spline curve, their tessellation algorithm can simulate the wrinkles. Oshita et al. [OSH 01] use a similar approach to Kang et al. These both methods are mainly applicable to flat surfaces where physical simulation can be done with a relatively small number (<1,000) of polygons. However, highly curved surfaces, such as sleeves, need to be simulated with a higher number of polygons.

Recently, Hadap et al. [HAD 99] have proposed to simulate the cloth wrinkles with bump map. The global movements of the mesh are simulated with a particle system. The bump/displacement map is the product of a modulation map and a wrinkling pattern. The modulation map is generated by computing the local deformation of cloth triangles and the wrinkling pattern is designed by the CG artist.

4.5 Example-based methods

Example-based approaches, also known as data-driven deformations have become popular since several years. They have been successfully used for motion data and other graphical objects such as deformable human face and body models. The main idea of example-based is to construct a simulator that can learn deformations through a set of examples. This simulator can be considered as a black box computing for each input parameter (user interaction) the corresponding output (the shape of the deformable object). During the training phase, a set of meaningful input/output data is given to the simulator, which learns. During the runtime execution, the simulator upon receiving an input computes the corresponding output. If the given input data corresponds exactly to one of the samples, the simulator will return the output of the sample. Otherwise, the returned output will be interpolated using one of the interpolation techniques such as RBF, k-nearest neighbours, neural network, or linear interpolation, etc.

Grzeszczuk et al. [GRZ 98] have used a neural network to animate dynamic objects. They have shown that physically based models can be replaced by a large neural network that automatically learns to simulate similar motions by observing the models in action. The neural network is trained with a set of samples that have been computed with the physically based method. Not surprisingly, their simulator works in real-time.

However, it has not been proven that the same method can be applied for complex simulation such as clothes.

Very recently, James et al. [JAM 03] have proven that the example-based approach can be successfully adapted to cloth simulation. The examples of cloth simulation they used had been calculated using commercial graphics package. Their approach makes use of Impulse Response Functions (IRF). Each IRF contains a sequence of *T* states (positions and velocity) defining an orbit of a cloth vertex, *T* being the time step of the simulation. To model a set of possible interactions, they construct a palette of IRF. Each IRF corresponds to a particular user interaction. During the runtime simulation, the orbit corresponding to the user interaction is played. If the user interaction is changed, a new orbit corresponding to the new user interaction is found and replaces the current one. To avoid discontinuity while changing the Impulse Response Function, they blend the two orbits. This work is interesting in the sense that it is the first method that aims to introduce example-based strategy in cloth simulation. However, this method is not adaptable to the simulation of clothes on virtual humans with its main drawback being the very limited user interaction.

Cordier et al. have presented several research works on real-time clothes [COR 02] [COR 03] [COR 04]. Their assumption is that the whole cloth does not need to be simulated with a general-purpose simulation method; instead many optimizations can be made. Their approaches work in a hybrid manner exploiting the merits of both the physical-based and geometric deformations. In addition, they make use of predetermined conditions between the cloth and the body model, avoiding complex collision detection and physical deformations wherever possible. These approaches will be described in greater detail in the next tutorial chapter.

5 Garment Design and Simulation

A Since the first developments to produce simulated garments on virtual characters [LAF 91] [CAR 92], cloth simulation and garment animation has made its way not only in computer research (Figure 3) [VOL 97], but also into commercial products aimed both for 3D computer design and the garment industry.

Two kinds of products ace currently available: Those oriented for general cloth simulation and animation, and those specialized for draping and fitting garment models on virtual mannequins. The first category offer tools for simulating any kind of deformable surface mechanically. They usually offer a simple mechanical model containing only the basic mechanical parameters of cloth (stiffness, viscosity, bending, and gravity) modelled as a spring-mass particle system and simulated using state-of-the-art integration techniques. They allow the computation of realistic cloth animation, but do not provide any tool for designing garments. The also offer general collision detection schemes for interaction with any other objects. These tools are usually integrated as plug-ins into 3D design and animation frameworks. Among the main products, there is MayaCloth integrated into Maya [ALI 04], Reactor [DIS 04], Stitch [DIG 04], SimCloth [CHA 04] and ClothReyes [REY 04] for 3D Studio Max, Dynamics integrated into Cinema 4D [MAX 04].

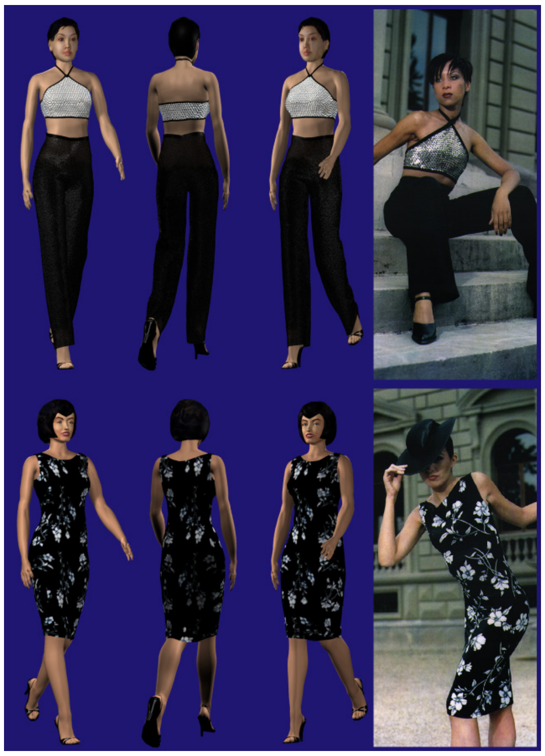


Figure 3: *Virtual fashion: Real models and their virtual counterparts.*

The second category focuses on garment draping on virtual mannequins for visualization (virtual fashion, web applications) and prototyping purposes (garment design applications). The CAD applications specialize the simulation on pattern assembly and garment draping using accurate mechanical models of fabrics, while the visualization application take advantage of geometric techniques for generating quickly realistic dressed mannequins out of design choices. Both use pattern models imported from professional pattern design tools (Gerber, Lectra, Investronica). These tools also usually provide a standalone environment for setting up the simulation and visualizing the results. Among them, there is Toyobo DressingSim [DRE 04], Browzwear V-Stitcher [BRO 04], or web applications such as FitMe.com [FIT 04] and MIRALab's Virtual Try-On [MIR 04] [COR 03].

Acknowledgements

This work is supported by the Swiss National Science Foundation (SNSF). Thanks to Marlène Arévalo and Christiane Luible for their design work to the garment models illustrating this chapter.

References

[AGU 90] T. Agui, Y. Nagao, and M. Nakajma, "An Expression Method of Cylindrical Cloth Objects-An Expression of Folds of a sleeve using Computer Graphics", Trans. of Soc. Of Electronics, Information and Communication, J73-D-II, 7, pp. 1095-1097, 1990.

[BAR 98] D. Baraff, A. Witkin, "Large Steps in Cloth Simulation", Computer Graphics (SIGGRAPH'98 proceedings), Addison-Wesley, 32, pp 106-117, 1998.

[BRE 94] D.E. Breen, D.H. House, M.J. Wozny, "Predicting the Drape of Woven Cloth Using Interacting Particles", Computer Graphics (SIGGRAPH'94 proceedings), Addison-Wesley, pp 365-372, July 1994.

[CAR 92] M. Carignan, Y. Yang, N. Magnenat-Thalmann, D. Thalmann, "Dressing Animated Synthetic Actors with Complex Deformable Clothes", Computer Graphics (SIGGRAPH'92 proceedings), Addison-Wesley, 26(2), pp 99-104, 1992.

[CHO 02] K.J. Choi, H.S. Ko, "Stable but Responsive Cloth", Computer Graphics (SIGGRAPH'02 proceedings), Addison Wesley, 2002.

[COL 91] J.R. Collier, B.J. Collier, G. O'Toole, S.M. Sargand, "Drape Prediction by means of Finite-Element Analysis", Journal of the Textile Institute, 82 (1), pp 96-107, 1991.

[COR 02] F. Cordier, N. Magnenat-Thalmann, "Real-time Animation of Dressed Virtual Humans", Eurographics 2002 Proceedings, Blackwell Publishers, 2002.

[COR 03] F. Cordier, H. Seo, N. Magnenat-Thalmann, "Made-to-Measure Technologies for Online Clothing Store", IEEE CG&A special issue on Web Graphics, IEEE Press, pp 38-48, 2003.

[COR 04] Cordier F., Magnenat-Thalmann N., "A Data-driven Approach for Real-Time Clothes Simulation", accepted to the 12th Pacific Conference on Computer Graphics and Applications (Pacific Graphics 2004), Cohen-Or, D., Ko, H.-S., Terzopoulos, D. and Warren, J., eds., October 6-8, 2004, Seoul Korea.

[DER 98] T. DeRose, M. Kass, T. Truong, "Subdivision Surfaces in Character Animation", Computer Graphics (SIGGRAPH'98 proceedings), Addison-Wesley, 32, pp 148-157, 1998.

[DEB 00] G. Debunne, M. Desbrun, M.P. Cani, A. Barr, "Adaptive simulation of soft bodies in real-time", Computer Animation, Annual Conference Series, IEEE Press, 2000.

[DES 99] M. Desbrun, P.Schröder, A. Barr, "Interactive Animation of Structured Deformable Objects", Proceedings of Graphics Interface, A K Peters, 1999.

[EBE 96] B. Eberhardt, A. Weber, W. Strasser, "A Fast, Flexible, Particle-System Model for Cloth Draping", Computer Graphics in Textiles and Apparel (IEEE Computer Graphics and Applications), IEEE Press, pp 52-59, Sept. 1996.

[EBE 00] B. Eberhardt, O. Etzmuss, M. Hauth, "Implicit-Explicit Schemes for Fast Animation with Particles Systems", Proceedings of the Eurographics workshop on Computer Animation and Simulation, Springer-Verlag, pp 137-151, 2000.

[EIS 96] J.W. Eischen, S. Deng, T.G. Clapp, "Finite-Element Modeling and Control of Flexible Fabric Parts", Computer Graphics in Textiles and Apparel (IEEE Computer Graphics and Applications), IEEE Press, pp 71-80, Sept. 1996.

[HAU 01] M. Hauth, O. Etzmuss, "A High Performance Solver for the Animation of Deformable Objects using Advanced Numerical Metds", Eurographics 2001 proceedings, Blackwell, 2001.

[HIN 90] B.K. Hind, J. McCartney, "Interactive Garment Design", Visual Computer, Springler-Verlag, Vol. 6, pp. 53-61, 1990.

[GAN 95] L. Gan et al, "A Study of Fabric Deformation using Non-Linear Finite Elements", Textile Research Journal, 65(11), pp 660-668, 1995.

[GRZ 98] R. Grzeszczuk, D. Terzopoulos, G. Hinton, "Neuroanimator: Fast neural network emulation and control of physics-based models", SIGGRAPH 98 Conference Proceedings, Annual Conference Series, pp. 9–20, Addison Wesley, 1998.

[HAD 99] S. Hadap, E. Bangarter, P. Volino, N. Magnenat-Thalmann, "Animating Wrinkles on Clothes", IEEE Visualization '99. San Francisco, USA, published by IEEE Press, pp. 175-182, Oct. 1999.

[JAM 99] D. James, D. Pai, "Accurate real-time deformable objects", SIGGRAPH 99 Conference Proceedings, Annual Conference Series, pp. 65–72, Addison Wesley, 1999.

[JAM 03] D. L. James and K. Fatahalian, "Precomputing Interactive Dynamic Deformable Scenes", ACM Transactions on Graphics (TOG), published by ACM SIGGRAPH Addison Wesley, Vol. 22(3), pp. 165-172, July 2003.

[KAN 00] Y.M. Kang, J.H. Choi, H.G. Cho, D.H. Lee, C.J. Park, "Real-Time Animation Technique for Flexible and Thin Objects", WSCG'2000 proceedings, pp 322-329, 2000.

[KAN 01] Y. M. Kang, J. H. Choi, H. G. Cho, D. H. Lee, "An efficient animation of wrinkled cloth with approximate implicit integration", The Visual Computer Journal, Springer-Verlag, 2001.

[KAN 02] Y.M. Kang, H.G. Cho, "Bilayered Approximate Integration for Rapid and Plausible Animation of Virtual Cloth with Realistic Wrinkles", Computer Animation 2000 proceedings, IEEE Computer Society, pp 203-211, 2002.

[LAF 91] B. Lafleur, N. Magnenat-Thalmann, D. Thalmann, "Cloth Animation with Self-Collision Detection", IFIP conference on Modeling in Computer Graphics proceedings, Springer-Verlag, pp 179-197, 1991.

[MEY 00] M. Meyer, G. Debunne, M. Desbrun, A. H. Barr, "Interactive Animation of Cloth-like Objects in Virtual Reality", Journal of Visualization and Computer Animation, John Wiley & Sons, 2000.

[NGG 95] H. Ng, R.L. Grimsdale, "GEOFF-A Geometrical Editor for Fold Formation", Lecture Notes in Computer Science Vol. 1024: Image Analysis Applications and Computer Graphic, R. Chin, et al., eds., Springer-Verlag, pp. 124-131, 1995.

[OSH 01] M. Oshita, A. Makinouchi, "Real-time Cloth Simulation with Sparse Particles and Curved Faces", Computer Animation, IEEE Press, 2001.

[PRE 92] W.H. Press, W.T. Vetterling, S.A. Teukolsky, B.P. Flannery, "Numerical Recipes in C", Second edition, Cambridge University Press, 1992.

[PRO 95] X. Provot, "Deformation Constraints in a Mass-Spring Model to Describe Rigide Cloth Behavior", Graphics Interface'95 proceedings, A K Peters Ltd, pp 147-154, 1995.

[SAK 91] Y. Sakagushi, M. Minoh, K. Ikeda, "A Dynamically Deformable Model of Dress", Trans. Society of Electronics, Information and Communications, pp 25-32, 1991.

[TER 87] D. Terzopoulos, J.C. Platt, H. Barr, "Elastically Deformable Models", Computer Graphics (SIGGRAPH'97 proceedings), Addison-Wesley, 21, pp 205-214, 1987.

[TER 88] D. Terzopoulos, K. Fleischer, "Modeling Inelastic Deformation: Viscoelasticity, Plasticity, Fracture", Computer Graphics (SIGGRAPH'88 proceedings), Addison-Wesley, 22, pp 269-278, 1988.

[VAS 01] T. Vassilev, B. Spanlang, "Fast Cloth Animation on Walking Avatars", Eurographics proceedings, Blackwell Publishers, 2001.

[VOL 95] P. Volino, M. Courchesne, N. Magnenat-Thalmann, "Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects", Computer Graphics (SIGGRAPH'95 proceedings), Addison-Wesley, pp 137-144, 1995.

[VOL 97] P. Volino, N. Magnenat-Thalmann, "Developing Simulation Techniques for an Interactive Clothing System", Virtual Systems and Multimedia (VSMM'97 proceedings), IEEE Press, Geneva, Switzerland, pp 109-118, 1997.

[VOL 00] P. Volino, N. Magnenat-Thalmann, "Implementing fast Cloth Simulation with Collision Response", Computer Graphics International Proceedings, IEEE Computer Society, pp 257-266, 2000.

[WEI 86] J. Weil, "The synthesis of Cloth Objects", Computer Graphics, SIGGRAPH 86 Conference Proceedings, Annual Conference Series, Vol. 20, pp. 49-54. Addison Wesley, 1986.

[YAN 93] Y. Yang, N.Magnenat-Thalmann, "An Improved Algorithm for Collision Detection in Cloth Animation with Human Body", Computer Graphics and Applications (Pacific Graphics'93 proceedings), World Scientific Publishing Co, 1, pp 237-251, 1993.

[GOV 03] N. Govindaraju, S. Redon, M. C. Lin, D. Manocha, "CULLIDE: interactive collision detection between complex models in large environments using graphics hardware", Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, published by ACM Press, pp. 25-32, 2003.

[BAC 03] Baciu G. and Wong S.K., "Image-based techniques in a hybrid collision detector", IEEE Transaction on Visualization and Computer Graphics, published by IEEE Press, 2002.

[MCN 99] McNeely A., Puterbaugh K. D., Troy J. J., "Six degree-of-freedom haptic rendering using voxel sampling", ACM Transactions on Graphics (TOG), published by ACM SIGGRAPH Addison Wesley, pp. 401-408, August 1999.

[ALI 04] http://www.aliaswavefront.com/

[DIS 04] http://www.discreet.com/

[DIG 04] http://www.digimation.com/

[CHA 04] http://www.chaosgroup.com/

[REY 04] http://www.reyes-infografica.com/

[MAX 04] http://www.maxon.de/

[DRE 04] http://www.dressingsim.com/

[BRO 04] http://www.browzwear.com/

[FIT 04] http://www.fitme.com/

[MIR 04] http://www.miralab.ch/

Physical Models and Numerical Solvers for Cloth Animations

Michael Keckeisen[1] Olaf Etzmuß[2] Michael Hauth[1]

[1]WSI/GRIS, University of Tübingen, Germany
[2]Indeed - Visual Concepts, Berlin, Germany

Abstract
Physical models and numerical solvers are the basis of current cloth simulation engines. Applications of cloth simulation range from pure visual effects for film and entertainment to demanding virtual try-on scenarios, where a high degree of physical realism combined with fast computation times is needed. Therefore, research in cloth animation has focused on improving realism as well as computation speed, and significant advances have been made over the last years.
In this part of the tutorial, we will discuss physical models for cloth ranging from simple mass-spring systems to continuum-based particle systems and fast finite element solutions. Then, we will describe several explicit and implicit time integration schemes and compare them with respect to stability and performance.

1. Introduction and Overview

The area of physically-based modeling is situated in the intersection of computer science, mathematics, and physics. The animation of cloth is a particularly interesting application of physically-based modeling, because it aims at fast animation solutions for rather difficult physical problems. Moreover, it addresses one of the major difficulties in creating realistic scenes with virtual actors.

The challenge of computer animation is to break down physical models for complex structures such as textiles, approximate them efficiently, and run fast simulations with intelligent numerical methods. The range of methods proposed in literature is quite large. The techniques vary from simplified methods designed specifically for real-time applications to sophisticated methods that were designed to reproduce measured material properties. Due to improved algorithms and faster computers, it becomes possible more and more to use advanced physical models and still achieve fast animations.

In this part of the tutorial, we will first discuss several physical models that have been employed for cloth animation in the past, ranging from discrete mass-spring and particle systems to finite element solutions for continuous cloth models. Then, we will explain explicit and implicit numerical methods for the solution of the arising ordinary differential equations.

2. Physical Models

Cloth models have been designed with different objectives. A common objective in computer graphics is to generate convincing pictures and films. For that purpose, physics may be ignored or simplified to a certain extent. A different objective is to preserve physical, measured properties in order to map real materials onto a simulated cloth. This, for instance, is indispensable in e-commerce applications, in which a customer selects clothes based on a simulation. In computer graphics, this also should lead to an animation that is fast and allows interaction with a complex scene.

In the following, we will start with relatively simple mass-spring and particle systems. After that we will describe how discrete and continuous models aim at preserving real material properties.

2.1. Discrete Models

All models which we take into account have in common that they discretize the cloth by a polygonal mesh (figure 1). The vertices of this mesh are called particles or (mass) nodes. In discrete models, the mesh topology defines, how the particles interact and exert forces on one another.

Given the mesh describing the cloth, forces on each particle are computed depending on its position and velocity, and the positions and velocities of a set of particles within

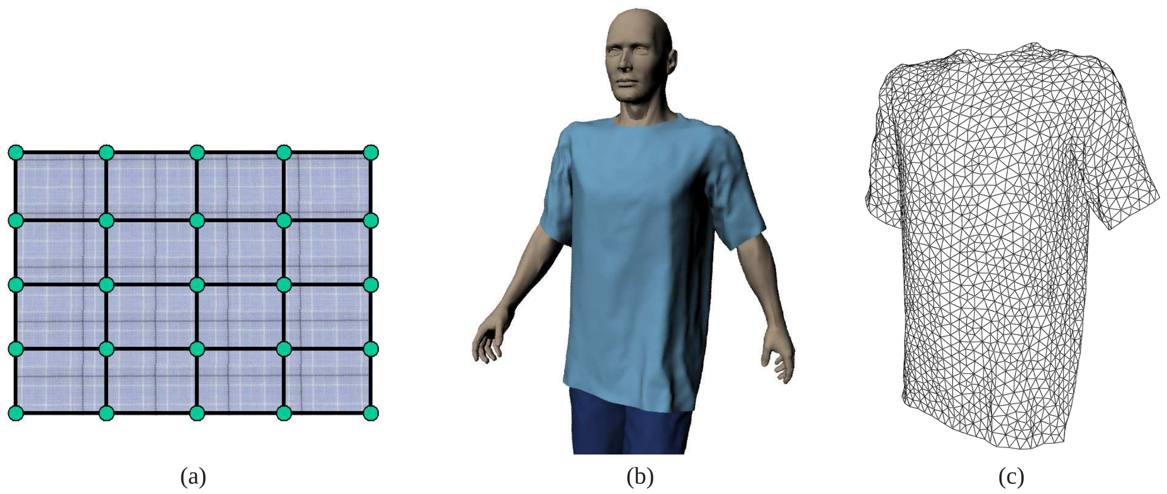


Figure 1: *Textiles are discretized by polygonal meshes. Image (a) shows a piece of cloth modelled as a quadrilateral mesh. The shirt in image (b) is represented by an unstructured triangle mesh (c).*

$\mathbf{s}(u,v)$	deformed surface
$\mathbf{r}(u,v)$	(local, partial) rest state of surface
$\mathbf{d}(u,v)$	displacement
$\mathbf{x}_i$	particle positions
$\mathbf{v}_i$	particle velocity
ε	strain (tensor)
σ	stress (tensor)
C	elastic tensor
D	viscous tensor
$\langle\mathbf{a},\mathbf{b}\rangle$	scalar product of vectors **a** and **b**
$\Delta\mathbf{s}$	Laplacian $\mathbf{s}_{xx}+\mathbf{s}_{yy}+\mathbf{s}_{zz}$
$\mathbf{s}_u$	partial derivative of **s** with respect to u

Table 1: *Notation in this section*

its topological neighbourhood. When the function F computing the forces has been determined, Newton's equation of motion governs the movement of the particles. The trajectory of each particle with mass m_i at position $\mathbf{x}_i$ is computed by

$$F(x,v) = m_i \cdot \frac{d^2\mathbf{x}_i}{dt^2}. \tag{1}$$

Here x denotes the vector containing all particle positions and v the vector of all particle velocities. Note that since particle systems already represent a discretization in space, only a system of ordinary differential equations has to be solved. The systems presented in literature differ by their methods of computing the forces.

2.1.1. Mass-spring systems

In mass-spring systems, particle interaction is solely modelled by linear springs.

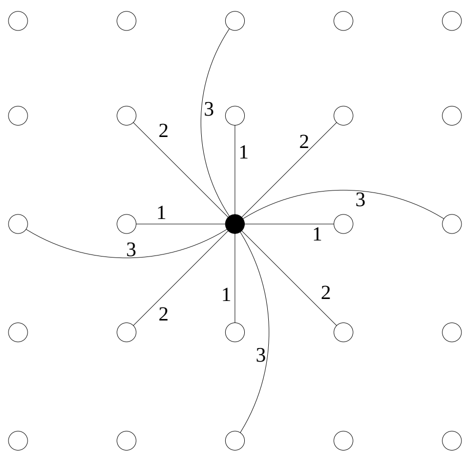


Figure 2: *Provot's mass-spring system with (1) structural springs, (2) shear springs, and (3) bending springs*

Provot [34] proposes a mass-spring system for textiles and uses a rectangular mesh in which the particles are connected by structural springs to counteract tension, diagonal springs for shearing, and interleaving springs for bending as shown in figure 2.

Forces by linear springs between two particles at $\mathbf{x}_i$ and

$\mathbf{x}_j$ are given by

$$F^e_{ij}(x) = k_{ij}(\|\mathbf{x}_i - \mathbf{x}_j\| - l_{ij})\frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|}, \qquad (2)$$

where k_{ij} is the elastic modulus of this spring and l_{ij} its rest length. The spring constant depends on the type of the spring. For the structural forces there are very large constants, whereas for the bend and shear forces the springs have small values.

Obviously, there is a strong interdependence between the different kinds of springs leading to nonlinear, uncontrolled effects. The diagonal shear springs, for instance, also lead to additional tension and transversal contraction.

Furthermore, we need viscous forces to account for energy dissipation due to internal friction. These forces damp out kinetic energy and depend on the velocity of the object. It is very popular to model these for each spring by

$$F^d_{ij}(x) = d_{ij}(\mathbf{v}_i - \mathbf{v}_j). \qquad (3)$$

Since these terms are linear they are particularly well suited for the numerical integration. However, there are two major disadvantages of this simple term as it also penalises a rigid rotation of a spring. Moreover, high damping of a structural spring prevents the object from bending. Hence, this simplified damping makes the deformable object move rather stiffly. These effects are alleviated by modelling a stiff, damped spring accurately by

$$F^d_{ij} = d_{ij}\frac{\langle \mathbf{v}_i - \mathbf{v}_j, \mathbf{x}_i - \mathbf{x}_j\rangle}{\|\mathbf{x}_i - \mathbf{x}_j\|^2}(\mathbf{x}_i - \mathbf{x}_j) \qquad (4)$$

This is just the linear damping term (3) projected onto the direction of the spring. Unfortunately, in many cases this term complicates the implicit time integration (cf. section 3.1.3).

Finally, in order to run the simulation, we only have to sum up all spring forces and plug them into equation (1).

In several mass-spring systems [34, 11, 28] another popular idea is exploited. It is motivated by a biphasic behaviour of textile materials as shown in figure 3 , i.e. initially the material yields to an exerted stress easily but appears to be extremely stiff in a second phase. This effect is imitated by rather small spring constants that model the first phase. In order to model the second, almost rigid phase, the system is post-processed after each time step if the springs are elongated too much. In this process, iteratively all particle positions are modified such that a certain maximum elongation is not exceeded. Such a post-processing is justified for simple mass-spring systems that do not model specific material properties anyway. Note that the result depends on the order in which the spring elongations are corrected.

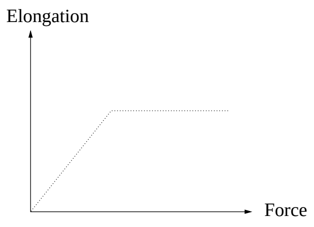


Figure 3: *Biphasic spring modelled by post-correction*

Although simple mass-spring systems do not model any specific material and are not related to measured properties of real clothes, they are capable of producing pleasing animations that are sufficient in many computer graphics applications. Visually very convincing animations based on a particle system with a sophisticated approach to handle bending behavior of cloth were presented in [6].

2.1.2. Representations of Cloth as discrete Mechanism

In their book on cloth modelling [26], Donald House and David Breen state that *"Cloth is a mechanism, not a continuous material"*. Consequently, some attempts have been made to model clothes by the interaction of discrete threads that are interwoven in textiles.

Some discrete systems that have been developed in computer animation for the animation of clothes and other surfaces have the advantage that they allow fast simulations. In particular, particle systems have been successfully used for rapid animations. We can consider the quadrilateral mesh that is described by the mass nodes and structural springs in Provot's mass-spring system as a network of interwoven threads, in which one thread is given by a chain of structural springs. Different threads can interact at the mass points, where shear, bend, or other internal forces apply. In order to model the interaction of threads, more complex forces than pure spring-forces are added to the system and yield a more general particle system.

Most particle systems use potential functions for tension, bend, and shear energy. These energies are chosen to correspond to standard experiments (Kawabata [29]) to measure textile properties. Hence, the measurements from one experiment are used to model one specific energy function. All energies are modelled on a rectangular grid, where each particle interacts with its four direct neighbours. The grid is aligned with two distinct directions that are apparent in textiles (in woven materials they are called weft and warp direction). The materials show different properties in these directions and each experiment has to be carried out for both directions.

The tension energy is evaluated for each particle and depends on the four neighbours of that particle in a rectangular mesh. The tension energy of a particle at position $\mathbf{x}_0$ is

$$E_t = \sum_{i=1}^{4} \begin{cases} \frac{1}{2}C_{t_1,i}(\|\mathbf{x}_0 - \mathbf{x}_i\| - l_i - h_{t_1,i})^3 & \text{if} \quad \|\mathbf{x}_0 - \mathbf{x}_i\| \geq l_i \\ \frac{1}{2}C_{t_2,i}(\|\mathbf{x}_0 - \mathbf{x}_i\| - l_i - h_{t_2,i})^5 & \text{if} \quad \|\mathbf{x}_0 - \mathbf{x}_i\| \leq l_i \end{cases}, \tag{5}$$

where l_i are the rest lengths between particles and $C_{t,i}$ and $h_{t,i}$ are material parameters. They can be used to fit measured data by a piecewise linear curve. The energy is computed from a strain $(\|\mathbf{x}_0 - \mathbf{x}_i\| - l_i)$, and the strain-stress relation is modelled piecewise cubic or quintic. If we introduce a linear strain-stress relationship by replacing the exponents with 2 and set $h_{t,i} := 0$, we get linear spring energies.

The shear energy is modelled as

$$E_s = \sum_{i=1}^{4} \frac{1}{2} C_s (\phi_i - \frac{\pi}{2} - h_{s,i})^2 \tag{6}$$

and the bend energy as

$$E_b = \sum_{i=1}^{2} \frac{1}{2} C_b (\psi_i - \pi - h_{b,i})^2. \tag{7}$$

Here C_s, C_b and $h_{s,i}, h_{b,i}$ are the material constants. These energies implement hinges functioning like springs that linearly depend on the shear angle ϕ and the bend angle ψ, respectively. These are the angles formed by the incident edges as depicted in figure 4.

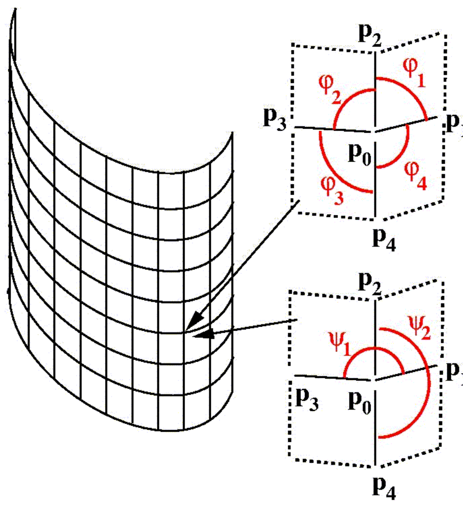


Figure 4: *Shear and bend energy in a particle system (image by Eberhardt [14])*

All derived energies are combined to compute the final forces to be plugged into equation (1):

$$F = -\text{grad}(E_t + E_s + E_b + E_{\text{external}}).$$

In this section, only elastic forces have been discussed. Viscous forces should be modelled in the fashion of section 2.2.7.

2.1.3. Triangular meshes

Until now, we only have considered particle systems based on rectangular meshes. Triangular mass-spring systems are widely used as well and can be constructed with almost the same set of forces. However, their physical properties are hard to control and depend on the topology of the mesh. Furthermore, they usually show a very strong transversal contraction. This motivated Volino [38, 40] to extend the concept of triangular mass-spring systems. In a triangular mesh the deformation of each face is uniquely determined by the elongation of its edges. Forces acting on each of its particles can be formulated depending only on these (vectorial) elongations. This results in a particle system in which the forces on one particle do not only depend on adjacent edges but on the elongations of all edges of all faces incident to the considered particle. The coefficients of these dependencies are the material constants and allow a flexible modelling of the physical properties.

The cloth simulation by Baraff and Witkin [2] also is based on a particle system for triangle meshes, however without the objective of fitting to real material data.

2.2. Continuous Models

Although clothes are not homogeneous, continuous objects, modelling them as discrete mechanism involves complications. As we cannot represent each single thread in a textile by an edge in the mesh, we have to choose a certain resolution of the object. If we want to be independent of this resolution, we need to represent a patch of textile as a continuous material, which allows us to use low resolution models without loosing basic material properties.

From a continuum model a consistent discretization can be derived. Consistency here means that the computed solution converges to the accurate solution for the continuum when the resolution is increased. That allows us to switch from one resolution to another without changing the properties of the cloth.

Therefore we will describe the foundations of the continuous theory, and present a particle system that can approximate this theory. Moreover, we describe a linear finite element solution for cloth modeling which provides about the same performance as particle systems if implicit time integration methods (section 3.1.3) are employed.

2.2.1. Descriptions of Strain

Continuum mechanics is the standard theory to describe and model deformable objects, and the following elaborations are based on several text books [4, 7, 31, 36, 3].

The basic quantities of continuum mechanics are *strain*, which is a dimensionless deformation noted by ε, and *stress*, which is a force per length for surfaces or per area for volumes and is denoted by σ. In the case of a one-dimensional spring, these entities are scalars. The strain of this spring is its elongation per length, while the stress is the spring force. In the case of surfaces or volumes, these entities are tensors.

Surfaces are more complicated than a one-dimensional spring, and the description of strain is more involved. Textiles can be described as regular surfaces (in the sense of differential geometry [12]). The deformation of a regular surface embedded in $\mathbf{R}^3$ is described by a strain tensor with respect to a certain undeformed reference state. In this equilibrium state, denoted by $\mathbf{r}$, the object is not deformed, and the elastic energy is zero. Let $\mathbf{r}$ be parametrised over a domain $U \times V$. Under forces the rest state deforms to a state $\mathbf{s}(u,v)$. The displacement is a mapping $\mathbf{d}$ defined by $\mathbf{d}(u,v) = \mathbf{s}(u,v) - \mathbf{r}(u,v)$ as depicted in figure 5.

The difference of the first fundamental forms I_s and I_r of the current state and the equilibrium state of the object describes the in-plane strain and defines a nonlinear strain tensor [30]

$$\tilde{G} = \tfrac{1}{2}(I_s - I_r) = \tag{8}$$
$$\tfrac{1}{2}\begin{pmatrix} \langle \mathbf{s}_u,\mathbf{s}_u\rangle & \langle \mathbf{s}_u,\mathbf{s}_v\rangle \\ \langle \mathbf{s}_u,\mathbf{s}_v\rangle & \langle \mathbf{s}_v,\mathbf{s}_v\rangle \end{pmatrix} - \tfrac{1}{2}\begin{pmatrix} \langle \mathbf{r}_u,\mathbf{r}_u\rangle & \langle \mathbf{r}_u,\mathbf{r}_v\rangle \\ \langle \mathbf{r}_u,\mathbf{r}_v\rangle & \langle \mathbf{r}_v,\mathbf{r}_v\rangle \end{pmatrix}.$$

For planar surfaces, the deformation is defined uniquely by the difference of the metrics of these states. As a piece of cloth is a surface in three-dimensional space, the curvature tensors (second fundamental forms) have to be taken into account as well. Terzopoulos and Fleischer [37] developed a model for animated surfaces based on the the energy due to these tensors.

Commonly, the rest state $\mathbf{r}$ is assumed to be the identity mapping. Then $\tilde{G}$ coincides with Green's strain tensor

$$G = \frac{1}{2}\begin{pmatrix} \langle \mathbf{s}_u,\mathbf{s}_u\rangle - 1 & \langle \mathbf{s}_u,\mathbf{s}_v\rangle \\ \langle \mathbf{s}_u,\mathbf{s}_v\rangle & \langle \mathbf{s}_v,\mathbf{s}_v\rangle - 1 \end{pmatrix}.$$

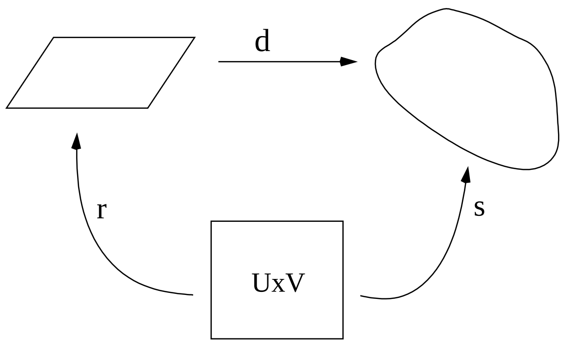


Figure 5: *The reference configuration: the rest state is parametrised by a mapping* $\mathbf{r}$ *on a space* $U \times V$. *By deformation* $\mathbf{d}$ *it transforms into the deformed (strained) configuration, which is parametrised by the mapping* $\mathbf{s}$.

Green's tensor, unfortunately, is nonlinear and yields fourth order terms in the energy formulation, and these are computationally very costly and lead to various numerical problems. Linear elasticity theory would allow much faster animations. It makes use of the linear approximation of Green's tensor, called Cauchy's strain tensor. It is obtained by neglecting terms of order higher than one in the displacement $\mathbf{d}$ in the components of Green's tensor, here for two dimensions:

$$\begin{aligned} \langle \mathbf{s}_u,\mathbf{s}_u\rangle - 1 &= \langle \mathbf{e}_1 + \mathbf{d}_u, \mathbf{e}_1 + \mathbf{d}_u\rangle - 1 \\ &= 2\langle \mathbf{d}_u, \mathbf{e}_1\rangle + O(\mathbf{d}^2) \\ \langle \mathbf{s}_v,\mathbf{s}_v\rangle - 1 &= \langle \mathbf{e}_2 + \mathbf{d}_v, \mathbf{e}_2 + \mathbf{d}_v\rangle - 1 \\ &= 2\langle \mathbf{d}_v, \mathbf{e}_2\rangle + O(\mathbf{d}^2) \\ \langle \mathbf{s}_u,\mathbf{s}_v\rangle &= \langle \mathbf{e}_1 + \mathbf{d}_u, \mathbf{e}_2 + \mathbf{d}_v\rangle \\ &= \langle \mathbf{d}_u, \mathbf{e}_2\rangle + \langle \mathbf{d}_v, \mathbf{e}_1\rangle + O(\mathbf{d}^2), \end{aligned} \tag{9}$$

where $(\mathbf{e}_1, \mathbf{e}_2)$ is the Cartesian basis of $\mathbf{R}^2$. Thus, Cauchy's tensor can be written as

$$\varepsilon = \begin{pmatrix} \mathbf{d}_u^1 & \frac{1}{2}(\mathbf{d}_v^1 + \mathbf{d}_u^2) \\ \frac{1}{2}(\mathbf{d}_v^1 + \mathbf{d}_u^2) & \mathbf{d}_v^2 \end{pmatrix},$$

where the superscripts denote the vector components. Linear elasticity is based on this linearised strain tensor and yields much simpler formulations. In particular, it results in linear partial differential equations. These linear equations are widely used in engineering and lend themselves to finite element formulations very easily.

The strain tensor of linear elasticity, as we have seen, is derived from Green's strain tensor by linearisations in the deformations, i.e. the displacement $\mathbf{d}$ (figure 5) is assumed to be small, and all terms of order two or higher in $\mathbf{d}$ are neglected in the strain tensor. For this reason the linear theory is not appropriate for highly flexible objects like clothes. It only applies to small displacements. It is therefore not invariant under rotations and leads to unphysical behaviour if the object or a part of it is rotated. As animated surfaces can bend strongly, the displacements become very large, although the deformations remain small. Thus, a linearisation with respect to a global reference frame is not applicable to cloth animation. However, we will see later (in sections 2.2.5 and 2.2.6) that a linearisation with respect to a local reference frame is feasible.

2.2.2. The equation of motion

So far, we only have dealt with the description of strain. In linear elasticity, also the relation between the stress tensor σ and strain ε is assumed to be linear, and the dependence is given by the elastic tensor C. This is formulated by Hooke's law:

$$\sigma_{ij} = C_{ijkl}\, \varepsilon_{kl} \tag{10}$$

C is a symmetric rank-4 tensor containing the material properties. Here, symmetry means $C_{ijkl} = C_{klij}$ as well as $C_{ijkl} =$

C_{jikl}. Hooke's law is used to compute the stress tensor in the equation of motion of a continuous elastic material:

$$\rho \frac{\partial^2 s}{\partial t^2} - \operatorname{div} \sigma = f, \tag{11}$$

where ρ is the mass density and the divergence of the stress σ yields the force density due to the interior energy of the elastic object. f denotes an external force density (e.g. the gravity force density ρg). Equation (11) is a partial differential equation (PDE) that has to be solved over the parameter domain and time. A standard procedure is to semidiscretize the system in space with finite differences or finite elements. This eliminates all spatial derivatives in the equation and reduces the PDE to an ordinary differential equation (ODE) in time t that can be solved by any suitable integration method. This way, we reduce the PDE (11) to the ODE (1).

2.2.3. Bend forces

Bend forces cannot be derived from the standard strain tensor because they are out-of-plane forces and arise only due to a volumetric property of the object (an ideal surface does not exhibit bending forces). Hence, there are basically two possibilities of deriving bending forces. First, we can simply add some bending forces to the in-plane forces, derived for instance from a Laplacian formulation or from a thin plate energy. Second, the cloth can be modelled as a thin volumetric object. In this case, shell theory provides the appropriate mechanism. Shells are thin objects which span over surfaces. The shell is parameterised by a mid-surface $\mathbf{s}$:

$$\mathbf{x}(u,v,w) = \mathbf{s}(u,v) + w\mathbf{d}(u,v),$$

i.e. the object position is described by the mid-surface parameters u, v and the height over the mid-surface w in the direction of a director $\mathbf{d}$ that has unit length.
From this description a three-dimensional strain and stress tensor is derived. Shells comprise (in-plane) membrane forces as well as bending forces. Some authors [15, 27, 19] model textiles and other thin deformable objects accurately by a nonlinear shell theory. In order to solve the resulting nonlinear PDE's, the system is discretized by finite elements, and Newton's method is used to compute an equilibrium solution.

2.2.4. Description of Clothes in a Linear Theory

A characteristic property of textiles is orthotropy. Linear, orthotropic materials possess two orthogonal symmetry axes in continuum mechanics. This reduces the number of free elastic material constants (entries in the elastic tensor) to four for in-plane deformations. The symmetry axes or directions in a woven material are the weft and warp directions. The textiles show very different physical behaviour in these directions, and this characterises the materials. Therefore, material measurements are carried out for the two directions independently [29]. The two Young moduli $k_1 := C_{1111}$ and $k_2 := C_{2222}$, the shear modulus $\mu := C_{1212} = C_{2121} = C_{2112} = C_{1221}$, and the parameter $C_{1122} = C_{2211}$, which controls the transverse contraction, have to be taken into account as elastic constants to model the in-plane stress. Additionally, the bending moduli B_1 and B_2 describe the curvature elasticity in the weft and warp directions.

Unfortunately, the strain/stress relation in textile materials is highly nonlinear, and approximations to these nonlinear properties have been an issue in computer animation. Generally, the nonlinear stress-strain relation is modelled by a piecewise linear function. This can be achieved by updating the the elastic tensor according to the current slope of the strain/stress curve [14]. The updating is carried out before each time step.

2.2.5. Continuous Theory and Particle Systems

As we have seen, Cauchy's stress tensor is not valid for large displacements. Green's strain tensor, however, impedes a rapid numerical solution and makes implicit time integration very difficult (see chapter 3).

In this section, we will show how the diagonal components of the strain tensor can be linearised locally to alleviate the solution of the ODE. From this model, we can derive a particle system that models continuous objects by a finite difference discretisation. This elastic model will be summarised here briefly. For details we refer to the full article [16].

Given the deformed surface $\mathbf{s}(u,v)$, we approximate the stress tensor by

$$\sigma = \begin{pmatrix} k_1(\|\mathbf{s}_u\| - 1) & \frac{1}{2}\mu < \mathbf{s}_u, \mathbf{s}_v > \\ \frac{1}{2}\mu < \mathbf{s}_u, \mathbf{s}_v > & k_2(\|\mathbf{s}_v\| - 1) \end{pmatrix}, \tag{12}$$

where k_1, k_2, and μ are the elastic material constants. This tensor is plugged into the equation of motion for continuous objects (11). In particular, in the strain tensor the dominating diagonal components have been linearised with respect to a local rest frame. Thus, the numerical solution is alleviated. From the diagonal components of the stress tensor we derive the following expression for the stress by finite difference discretization:

$$\sigma_{ii} = \frac{k_{1,2}}{l^2}(\|\mathbf{x}_i - \mathbf{x}_j\| - l),$$

where l is the rest distance between the particles at $\mathbf{x}_i$ and $\mathbf{x}_j$. Linear spring forces result from approximating the divergence of the diagonal entries in the stress tensor

$$f_{i,j} = \frac{k_{1,2}}{l^2}\left[(\mathbf{x}_i - \mathbf{x}_j) - l\frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|}\right], \tag{13}$$

Note that the $f_{i,j}$ are actually force densities as we have continuous objects.

Additionally to the stretch forces, shear forces can be approximated similarly, and bend forces can be derived from a thin plate energy (see [16]).

For rectangular meshes it can be verified that this particle system approximates the accurate continuum model with linear precision, i.e. we have convergence to the accurate solution when the resolution is increased. For general quadrilateral meshes, the solution still converges, but does not possess the approximation property.

2.2.6. A Linearised Finite Element Model

The approach described in the last section works only for regular quadrilateral meshes. Therefore, a linearised finite element model for cloth which works for arbitrary unstructured triangle meshes has been developed [17]. It is particularly designed for numerically stiff materials such as textiles because it yields linear equations in each time step and allows fast time stepping in an implicit integration method (cf. section 3.1.3).

For the moment we assume that all forces and displacements are in the (x,y)-plane, and that the small displacement assumption holds. For this case, we state the finite element formulation for linear elasticity as presented in several text books (e.g. [3]). Given the surface $\mathbf{s}(u_1,u_2)$, we denote the linear form functions of the linear finite element space by $N_j(u_1,u_2)$. For each triangle $T_m(a,b,c)$ we compute the form factors $\frac{\partial N_j}{\partial u_i}, j=a,b,c, i=1,2$ in the rest state (these form factors replace the rest lengths of the springs in a mass-spring system). Also we compute the triangle areas Ω_{T_m} in the rest state. Then we construct the stiffness matrix as follows. For each triangle T_m and each vertex pair (a,b) of this triangle, we define a submatrix $K^m_{ab} \in \mathbb{R}^{2\times 2}$

$$\left[K^m_{ab}\right]_{ij} = \left(\sum_{k,l=1}^{2} \frac{\partial N_a}{\partial u_i} C_{ikjl} \frac{\partial N_b}{\partial u_j}\right) \Omega_{T_m}, \qquad (14)$$

where C is the elastic tensor that contains all elastic material properties.

The stiffness matrix $A \in \mathbb{R}^{3n\times 3n}$, where n is the number of vertices, is assembled from all matrices K^m_{ab} for all triangles. It consists of $n \times n$ submatrices that are computed as follows:

$$[A]_{ab} = \begin{pmatrix} \left[\sum_{m|T_m\in\Gamma(a,b)} K^m_{ab}\right] & \begin{matrix}0\\0\end{matrix} \\ \begin{matrix}0 & 0\end{matrix} & 0 \end{pmatrix} \in \mathbb{R}^{3\times 3}$$

Here $\Gamma(a,b)$ is the set of all triangles containing vertices a,b.

We denote the displacement, position, and velocity vectors that store the values of all n vertices by $d,x,v \in \mathbb{R}^{3n}$, respectively. The force vector is then computed by

$$F(x) = Ad = A(x-\bar{x}),$$

where x is the vector of the vertex positions in the deformed state and $\bar{x}$ the vector of positions in the rest state. This force formulation is valid as long as the surfaces remain in the (x,y)-plane.

As mentioned previously, large rotations prevent the usage of the linear strain formulation. Hence, in each time step we are going to remove these rotations before computing the deformation forces (similar ideas have been used for three-dimensional deformable objects in [33] and [25]). This is equivalent to constructing a rotated rest state and linearising with respect to this rest state. The key is the polar decomposition of the deformation gradient. The deformation gradient J can be decomposed into a rotation R and a pure deformation U

$$J = R\cdot U.$$

The polar decomposition is carried out as follows:

1. Compute $U^2 = J^tJ$
2. Compute the eigenvalues λ_1,λ_2 and eigenvectors $\mathbf{v}_1,\mathbf{v}_2$ of U^2
3. Compute $U = \sqrt{\lambda_1}\mathbf{v}_1\mathbf{v}_1^t + \sqrt{\lambda_2}\mathbf{v}_2\mathbf{v}_2^t$
4. Compute $R = JU^{-1}$

Since we use linear finite elements, the matrix J is constant on each triangle. Hence, the polar decomposition gives us a unique rotation R^m for each triangle T_m. This rotation is used to move the deformed triangle back to a planar state in the (x,y)-plane, compute the elastic forces with linear elasticity there, and finally rotate the computed forces back in 3D space. For that purpose, we have to transform the displacement vectors before the multiplication with the local stiffness matrix. The force vector that is the result of the multiplication has to undergo the inverse transformation. One local stiffness matrix for edge (a,b) computes forces as follows:

$$F(x) \ = \ R^m K^m_{ab} (R^m)^t \mathbf{x}_a - R^m K^m_{ab} \bar{\mathbf{x}}_a$$

Note that the planar rest state position $\bar{\mathbf{x}}_a$ is already in 2D space and only the results of the multiplication need to be rotated to 3D space. Hence, we define the transformed local stiffness matrices as

$$\begin{aligned} Q^m &= R^m \cdot K^m \cdot R^t_m, \\ \bar{Q}^m &= R^m \cdot K^m, \end{aligned}$$

where $R^m \in \mathbb{R}^{3\times 2}, Q^m \in \mathbb{R}^{3\times 3}, \bar{Q}^m \in \mathbb{R}^{3\times 2}$ The global stiffness matrices are assembled from the matrices Q^m and $\bar{Q}^m$:

$$[A]_{ab} = \sum_{m|T_m\in\Gamma(a,b)} Q^m_{ab} \qquad (15)$$

$$[\bar{A}]_{ab} = \sum_{m|T_m\in\Gamma(a,b)} \bar{Q}^m_{ab}$$

The global force vector is computed by

$$F(x) = Ax - \bar{A}\bar{x}.$$

Similarly, bend forces can be derived from a finite element formulation of the Laplacian operator projected onto the surface normals. Then, they can be evaluated as

$$F(x) = \tilde{A}x.$$

Moreover, linear viscous forces can be derived by replacing the elastic tensor C by the viscosity tensor D. These viscous forces are evaluated by

$$F(v) = Bv.$$

For a detailed discussion of bend forces and viscosity we refer to [17]. Summarizing all forces, Newton's equation of motion becomes

$$M\frac{d^2x}{dt^2} = Ax - \overline{A\overline{x}} + \tilde{A}x + Bv + f_{\text{ext}},$$

where M is the mass matrix, and f_{ext} are external forces. One time step in the animation is then computed as follows:

1. Compute the rotations R^m for each triangle T_m
2. Construct the matrices $A, \tilde{A}, B$ and compute $\overline{A\overline{x}}$
3. Carry out a time step.

The achievement of this procedure is that the forces are linear, such that a system of linear equations has to be solved in each time step, if an implicit time integration method is used. For the implicit Euler method (see section 3.1.3), for instance, we have to solve

$$\begin{aligned} v_1 &= v_0 + hM^{-1}(Ax_1 + \tilde{A}x_1 + Bv_1 - \overline{A\overline{x}} + f_{\text{ext}}) \\ x_1 &= x_0 + hv_1 \end{aligned}$$

with time step h. Note that the subscripts denote the time indices here. From this, we obtain the linear system

$$(M - h^2A - h^2\tilde{A} - hB)v_1 = Mv_0 + h(Ax_0 + \tilde{A}x_0 - \overline{A\overline{x}} + f_{\text{ext}}).$$

2.2.7. Viscosity and Hysteresis in Textiles

Figure 6 shows a typical strain/stress curve that can be measured when a textile patch is stretched and relaxed. Hysteresis effects as in this figure are due to energy dissipation. When the material deforms under external forces, the strain/stress relation is described by the upper branch of the hysteresis. When it is released and contracts again, this relationship is described by the lower branch and the area between both branches is proportional to the dissipated energy.

There are several causes for energy dissipation when modelling deformable objects. Here, we focus exclusively on intrinsic effects, i.e. those that do not depend directly on external forces like friction due to fluid flow (wind). Internal friction depends on the relative motion of parts of the deformable object, that is, on the strain rate. The strain rate tensor is defined as

$$\nu_{ij} = \frac{d\varepsilon_{ij}}{dt}. \tag{16}$$

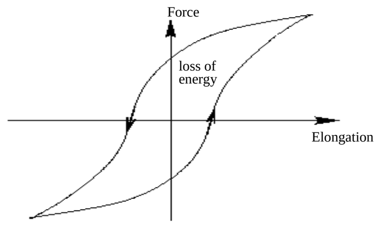


Figure 6: *Hysteresis*

The forces generated by internal friction are often called viscous in analogy to Newtonian fluids. A viscous stress can be computed analogously to equation (10):

$$\sigma^v_{ij} = D_{ijkl}\,\nu_{kl}, \tag{17}$$

where the "viscosity tensor" D has the same structure as the elastic tensor C. The viscous stress is added to the purely elastic stress. In most implementations the viscosity tensor D is chosen constant and proportional to the elastic tensor C. This produces a material called a Kelvin-Voigt solid, the simplest viscoelastic solid. Note that the spring damping forces as stated in equation (4) can be derived from this model. In order to fit the typical behavior of cloth, a more complex and visco-elastic model has to be chosen. For instance a constant Q model allows to reproduce a measured hysteresis curve [20].

3. Numerical Simulation

In this section, we will describe several explicit and implicit time integration methods, in parts following Hauth [23]. A further in-depth comparison of integration methods for cloth simulations has been presented by Volino and Magnenat-Thalmann [41].

3.1. Methods for Numerical Integration

As we have seen in the previous section mechanical systems are often given as a second order ordinary differential equation accompanied by initial values

$$\begin{aligned} x''(t) &= f_v\big(t, x(t), x'(t)\big), \\ x(t_0) &= x_0, x'(t_0) = v_0. \end{aligned} \tag{18}$$

The differential equation can be transformed into a first order system by introducing velocities as a separate variable:

$$\begin{bmatrix} x(t) \\ v(t) \end{bmatrix}' = \begin{bmatrix} v(t) \\ f_v\big(t, x(t), v(t)\big) \end{bmatrix}, \quad \begin{bmatrix} x(t_0) \\ v(t_0) \end{bmatrix} = \begin{bmatrix} x_0 \\ v_0 \end{bmatrix}. \tag{19}$$

For the next few sections it will be convenient to write this ODE in the more abstract form

$$y'(t) = f\big(t, y(t)\big), \quad y(t_0) = y_0, \tag{20}$$

before we will come back to the special setting (19) for eventually gaining computational advantages.

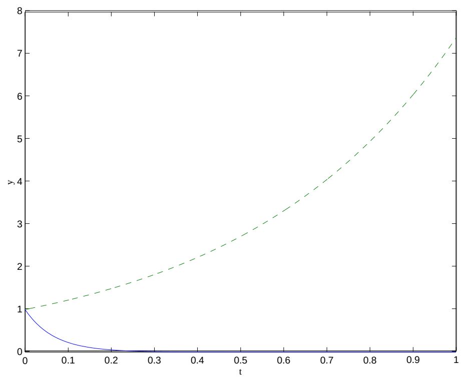


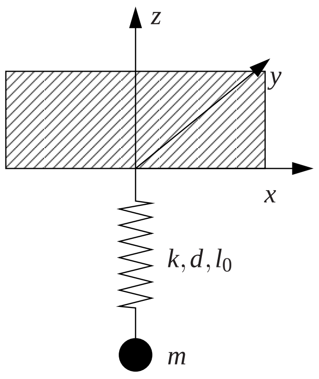
Figure 7: *Solutions of example 1 for* $\lambda = 2$ *(dashed) and* $\lambda = -15$ *(solid)*

Throughout the following discussion we will use the following examples:

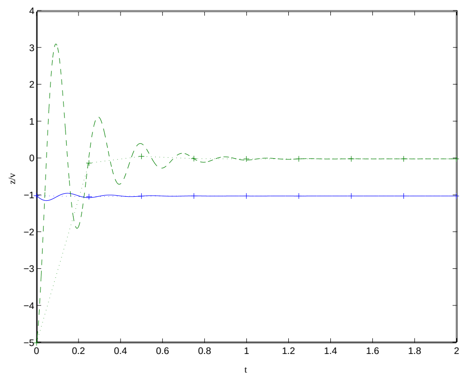
1. $y' = \lambda y$, $y(0) = 1$ with $\lambda = 2, -15$ for $t \in [0,1]$ (figure 7).
2. The overdamped wave equation $y'' = \lambda/2y + \lambda y'$ with $\lambda = 5$ for $t \in [0,10]$ and starting values $y(0) = 0, y'(0) = 1$. It has the analytical solution $y(t) = 1/15\sqrt{15}e^{1/2\left(-5+\sqrt{15}\right)t} - 1/15\sqrt{15}e^{-1/2\left(5+\sqrt{15}\right)t}$
3. This example is based on a simple mechanical system (figure 8(a)): A particle p with mass m connected to the origin using a spring with stiffness k, damping d and rest length l_0, is pulled down by gravity. This setting is described by the ODE

$$\frac{d^2z}{dt^2} = \frac{k\,(l_0 - z)}{m} - \frac{d}{m}\frac{dz}{dt} + g_z. \qquad (21)$$

We set the parameters $m = 0.1$, $k = 100$, $l_0 = -1$, $d = 1$, $g_z = -10$, $z_0 = -1$, $v_{0,z} = -5$ and simulate the interval $t \in [0,2]$ (figure 8(b)).



(a) The mechanical system of example 3.



(b) Exact solution of example 3: z (solid), z' (dashed), and an implicit Euler solution (see below) for large timesteps.

Figure 8: *Example 3.*

3.1.1. Explicit methods

The oldest and most simple method of integration is the so called forward or explicit Euler method. Time is discretised into slices of length h. To get a formula for advancing a timestep, h the differential quotient on the left hand side of (20) is replaced by the forward difference quotient

$$\frac{y(t+h) - y(t)}{h} \approx y'(t) = f(t, y(t)). \qquad (22)$$

Thus we get the integration formula for advancing a single timestep

$$y(t+h) = y(t) + hf(t, y(t)). \qquad (23)$$

Iterating this method gives a sequence of numerical approximations Y_n. Geometrically this method can be interpreted as straightly following the tangent of the solution and then recalculating the slope for the next step.

There are several criteria for evaluating an integration method:

- convergence
- accuracy
- stability
- efficiency

Convergence means that for $h \to 0$ the numerical solutions Y_n meet the analytical. All useful methods must be convergent, so we won't discuss non-convergent methods or criteria for convergence. More interesting is the accuracy or order of a method. By this we mean how fast a method converges for $h \to 0$, or with other words how accurate the solution is for a given h. By using a taylor expansion for the exact solution after a single timestep

$$y(t+h) = y(t) + hy'(t) + h^2/2y''(t) + O(h^3) \qquad (24)$$

we find that for the numerical approximation Y_1 produced by an explicit Euler step

$$y(t_1) - Y_1 = O(h^2). \qquad (25)$$

If we continue the method using the *numerical solution* Y_1 as a starting value for the next timestep we lose[21] a power of h for the global error

$$y(t_n) - Y_n = O(h). \qquad (26)$$

This means that the explicit Euler method converges linearly or has *order* 1. We will analyse stability and efficiency of the method later.

As a next step we will introduce methods of higher order. For this a centered difference estimation for $y'(t+h/2)$ (20) is used

$$\frac{y(t+h) - y(t)}{h} \approx y'(t+h/2) = f(t, y(t+h/2)). \qquad (27)$$

and thus as an iteration scheme

$$Y_{n+1} = Y_n + f(t, y(t_n + h/2)). \qquad (28)$$

But how do we find $y(t_n + h/2)$? For an estimation we use an explicit Euler step to get

$$k_1 = Y_n + \tfrac{h}{2} f(t, Y_n) \qquad (29)$$

$$Y_{n+1} = Y_n + hk_1, \qquad (30)$$

the so called *explicit midpoint* rule. The estimation by forward Euler, although not very accurate is good enough, as the function evaluation is multiplied by the timestep to advance to the next approximation. So by a taylor expansion we find a local error of $O(h^3)$ leading to a global error of

$$Y_n - y(t_n) = O(h^2) \qquad (31)$$

for the explicit midpoint rule.

Generalizing the idea of using function evaluations at s intermediate points $t + c_j h$ leads to the Runge-Kutta methods defined by a Runge-Kutta matrix (a_{ij}), weights b_i, abscissae c_j and the equations

$$k_i = Y_n + h \sum_{j=1}^{s} a_{ij} k'_j$$

$$\text{with } k'_i = f(t_n + c_i h, k_i) \text{ for } \quad i = 1, \ldots, s$$

$$Y_{n+1} = Y_n + h \sum_{i=1}^{s} b_i k'_i \qquad (32)$$

The coefficient set can comfortably be specified like in table 2. If the matrix (a_{ij}) is strictly upper, all inner stages k_i only depend on k_j with $j < i$ and thus can be computed one after the other.

The most famous one is the method by Runge and Kutta given in table (3) together with the explicit midpoint rule

c_1	a_{11}	a_{12}	$\cdots$	a_{1s}
c_2	a_{12}	a_{22}	$\cdots$	a_{2s}
$\vdots$	$\vdots$		$\ddots$	$\vdots$
c_s	a_{s1}	a_{s2}	$\cdots$	a_{ss}
	b_1	b_2	$\cdots$	b_s

Table 2: *General Runge–Kutta method*

interpreted as a Runge-Kutta method. The method by Runge and Kutta possesses order 4.

0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	0
	0	1

0				
1/2	1/2			
1/2	0	1/2		
1	0	0	1	
	1/6	2/6	2/6	1/6

Table 3: *Explicit midpoint and "the" Runge-Kutta method*

By using algebraic relations for the coefficients, it is possible to construct explicit Runge-Kutta methods of arbitrary high order resulting in many inner stages with many function evaluations. But for most practical applications order 4 is sufficient.

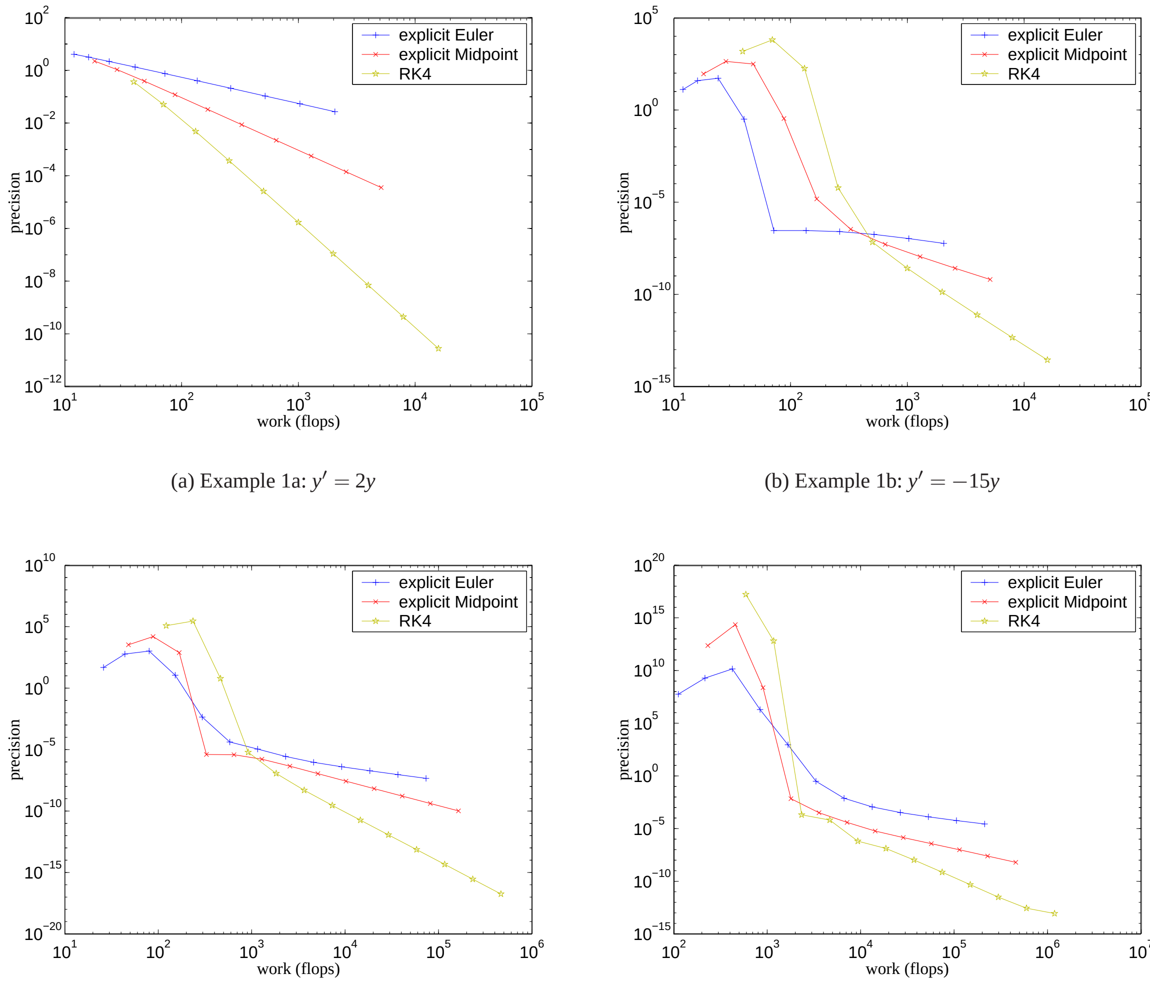
Having constructed all these methods, we now will try them on our examples.

The diagrams in figure 9 were produced by solving the examples using different timesteps and measuring the number of floating point operations needed for achieving the specified accuracy when compared with the (analytical) reference solution. In the work-precision diagram the y-axis shows the error $\|Y_{t_{end}} - y(t_end)\|$ as a function of the required number of floating point operations. The first example with $\lambda = 2$ (figure 9(a)) shows exactly the expected behaviour: when reducing the time step and thus investing more work, the numerical solutions converge against the reference solution. Moreover the slope of the curves in the double logarithmic plot exactly match the order of the method. But in all the other examples (figure 9(b)-9(d)) this behaviour only shows up after an initial phase, where the solver produces completely wrong results. This is the point where stability comes in. We will now analyse this by using the simplest example where it occurs, i.e. example 1 with $\lambda < 0$.

3.1.2. Stability

The equation for example 1 is called *Dahlquist's test equation*

$$y' = \lambda y, \qquad \lambda \in \mathbb{C}. \qquad (33)$$



(a) Example 1a: $y' = 2y$

(b) Example 1b: $y' = -15y$

(c) Example 2: $y'' = -5y' - 5y$

(d) Example 3

Figure 9: *Work precision diagrams for the explicit Euler, explicit midpoint and RK4 methods.*

Its exact solution to an initial value $y(0) = y_0$ is given by

$$y(t) = e^{\lambda t} y_0. \tag{34}$$

This equation is a tool for understanding and evaluating the stability of integration methods. We have seen, that in the damped case characterised by $\mathrm{Re}\,\lambda < 0$ convergence is only achieved by very small timesteps. In this case, since the exponent is negative, the analytical solution is bounded for $t \longrightarrow \infty$. Therefore a meaningful numerical method is required to deliver a bounded solution. An integration scheme that yields a bounded solution is called *stable*.

If we apply the explicit Euler's method with a fixed step size h to (33) the n-th point of each solution is given by:

$$Y_n = (1 + h\lambda)^n y_0 \tag{35}$$

For the explicit Euler scheme the numerical solution given by (35) is bounded if and only if $|1 + h\lambda| < 1$, i.e. for $h\lambda$ in the unit ball around -1. A similar analysis can be carried out for the other methods and also results in restrictions of the admissible step size.

This analysis explains the sharp bend in figures 9(b)-9(d). Only when the step size drops below a certain limit dictated by λ (by $h < \lambda^{-1}$ in case of the forward Euler method) the numerical solutions can converge. If the damping is increased, i.e. $\mathrm{Re}\,\lambda \rightarrow -\infty$, then for the explicit Euler necessarily $h \longrightarrow 0$ for the solution to be stable. This means the step size is artificially limited and it cannot be increased beyond the stability limit in order to save work sacrificing some - but not all - accuracy.

3.1.3. The implicit Euler method

To construct a method that better suits our needs we go back to (20) and substitute the differential quotient by a *backward* difference quotient for $y(t+h)$

$$\frac{y(t+h)-y(t)}{h} \approx y'(t+h) = f(t+h, y(t+h)). \tag{36}$$

This time this results in the integration formula

$$Y_{n+1} = Y_n + hf(t+h, Y_{n+1}), \tag{37}$$

the so called backward or implicit Euler method. As its explicit variant this method can be shown to have order 1. Now the numerical solution only is given implicitly as the solution of the possibly nonlinear equation

$$Y_{n+1} - hf(t+h, Y_{n+1}) - Y_n = 0. \tag{38}$$

If we apply this method to the Dahlquist equation we get the recurrence formula

$$Y_n = (1-h\lambda)^{-n} y_0. \tag{39}$$

The numerical solution Y_n remains bounded for $|(1-h\lambda)^{-1}| < 1$. If we assume $\lambda < 0$, this holds for arbitrary $h > 0$. Thus there is no restriction on stepsize, the method is *unconditionally stable*. Figure 11 shows the work-precision diagrams for the implicit Euler method and our examples. We observe that we never loose stability and we especially do not miss the solution by several orders of magnitude compared to the explicit methods, although we loose some accuracy when the timesteps become large.

As a practical tool for graphically visualizing the stability properties of a method we define the *stability region* $\mathcal{S}$ to be the set of parameters, for which the integration method yields a bounded solution:

$$\begin{aligned}\mathcal{S} := \{z := h\lambda \in \mathbb{C} : \ &\text{the numerical integration}\\ &\text{of equation (33) with step size } h \text{ and}\\ &\text{parameter } \lambda \text{ is stable}\}.\end{aligned} \tag{40}$$

Methods that contain the complete left half-plane in $\mathcal{S}$ are called *A-stable* or *unconditionally stable*. They are well suited for the stable integration of stiff equations. Obviously, the implicit Euler scheme is A–stable, whereas its explicit counterpart is not. The stability regions of all presented methods are shown in figure 10.

After reviewing the process that led us to the definition of the stability region, we can outline a more general idea that will allow us to determine easily the stability of more complex methods. The idea for analysing both Euler methods applied to (33) was to find a closed expression describing the *stability function* $\mathcal{R}$. This function maps the initial value y_0 to the value Y_1, performing a single step of the method

$$\mathcal{R} : y_0 \mapsto Y_1. \tag{41}$$

Thus $Y_n = R(h\lambda)^n y_0$. For the explicit Euler method we found in (35)

$$\mathcal{R}(z) = 1 + z, \tag{42}$$

for the implicit version in (39)

$$\mathcal{R}(z) = \frac{1}{1-z}. \tag{43}$$

The definition for the stability region now reads

$$\mathcal{S} = \{z \in \mathbb{C} : |\mathcal{R}(z)| < 1\}. \tag{44}$$

3.1.4. Methods of higher order

To find a higher order method, we go back to equation (28) and do not replace the $y(t+h/2)$ but take the formula as an implicit definition of $y(t+h)$. We get the *implicit midpoint rule*

$$Y_1 = Y_0 + hf\left(\frac{Y_1 + Y_0}{2}\right). \tag{45}$$

Further on we use a simplified notation for advancing one step, writing Y_0 and Y_1 instead of Y_n and Y_{n+1}.

Alternatively the midpoint rule can be derived as a *collocation method* with $s = 1$ internal nodes, i.e. by constructing a polynomial interpolating the particle trajectories at a given, fixed set of s nodes[22]. This idea allows for the construction of implicit Runge-Kutta methods with arbitrary order. In contrast to explicit methods the matrix (a_{ij}) ceases to be strictly lower triangular. These methods are computationally more expensive, so we just stick to the midpoint rule. Its stability function is given by

$$\mathcal{R} = \frac{1+z/2}{1-z/2}. \tag{46}$$

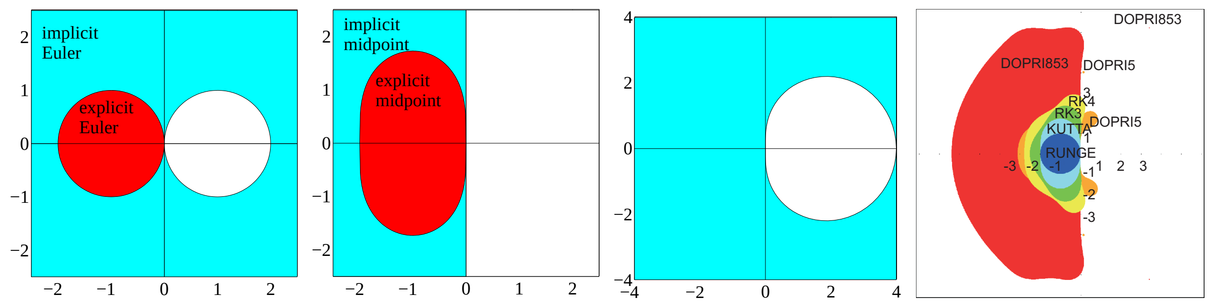
As $\mathcal{R} \le 1$ for any $\operatorname{Re} z < 0$ the implicit midpoint rule is *A*-stable.

As another possible choice we now introduce *multistep methods*. They are computationally inexpensive because they have no inner stages and some of them are A-stable. A multistep method with k steps is of the general form

$$\sum_{j=0}^{k} \alpha_j Y_{k-j+1} = h \sum_{j=0}^{k} \beta_j f_{k-j+1}, \tag{47}$$

with $f_{n+j} := f(t_{n+j}, Y_{n+j})$. Here we also have ‘history points’ with negative indices. The coefficient α_k is required to be nonzero; for variable time step sizes the coefficients depend on the last stepsizes, which we have omitted here for the ease of demonstration. Important special cases are the class of *Adams methods* where $\alpha_0 = \cdots = \alpha_{k-2} = 0$:

$$Y_1 = Y_0 + h \sum_{j=0}^{k} \beta_j f_{k-j+1} \tag{48}$$



(a) Euler methods (b) midpoint rules (c) BDF(2) (d) RK methods

Figure 10: *Stability regions (shaded) of the methods.*

and the class of *BDF–methods* (**b**ackward **d**ifferentiation **f**ormulas) with $\beta_0 = \cdots = \beta_{k-1} = 0$:

$$\sum_{j=0}^{k} \alpha_j Y_{k-j+1} = h\beta_k f_1. \tag{49}$$

If the formula involves the right-hand side f_1 at the new approximation point Y_1 the method is said to be implicit. BDF-methods are always implicit. The coefficients can again be constructed by a collocation approach. The stability regions of implicit and explicit Adams methods are bounded and located around the origin, thus they are not interesting for large time steps.
BDF-methods were the first to be developed to deal with stiff equations and possess an unbounded stability region covering a sector within the negative complex half-plane. Therefore they are widely used today. For $k+1$ points, these methods possess order $k+1$ and only one nonlinear system has to be solved, whereas s coupled systems have to be solved for an s-stage implicit Runge-Kutta method.

The BDF-method for $k = 1$ is just the implicit Euler method, for k=2 the method is given as

$$Y_1 = \frac{4}{3}Y_0 - \frac{1}{3}Y_{-1} + \frac{2}{3}hf(t+h, Y_1) \tag{50}$$

The stability region of BDF(2) and the other implicit methods are shown in figure 10.

3.1.5. The Verlet method

As a last method we will discuss a scheme commonly referred to as leapfrog or Stoermer-Verlet method. It is especially efficient if (18) is given as the second order system

$$x''(t) = f_v\big(t, x(t)\big), \tag{51}$$

i.e. $f_v\big(t, x(t), x'(t)\big) = f_v\big(t, x(t)\big)$. It is not applicable to general first order systems of the form (20).

To derive it, we use centered differences at a staggered

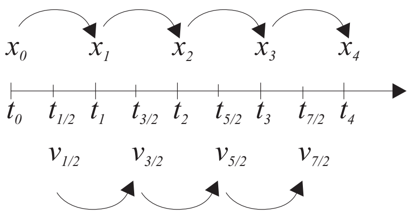


Figure 12: *Staggered grids for the Verlet method.*

grid (figure 12) i.e. we now approximate v at $t+(2i+1)h/2$ and x at $t+ih$ by centered differences

$$\frac{v_{n+1/2} - v_{n-1/2}}{h} = f(x_n) \tag{52}$$

$$\frac{x_{n+1} - x_n}{h} = v_{n+1/2} \tag{53}$$

thus

$$v_{n+1/2} = v_{n-1/2} + hf(x_n) \tag{54}$$

$$x_{n+1} = x_n + hv_{n+1/2}. \tag{55}$$

The method possesses order 2 as one can see by substituting (54) into (55) resulting in the second order centered difference

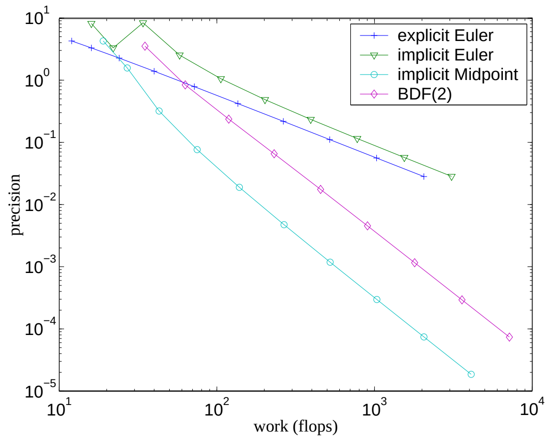
$$\frac{x_{n+1} - 2x_n + x_{n-1}}{h} = f(x_n). \tag{56}$$

From this equation an alternative formulation of the Verlet scheme as a multistep method can be derived

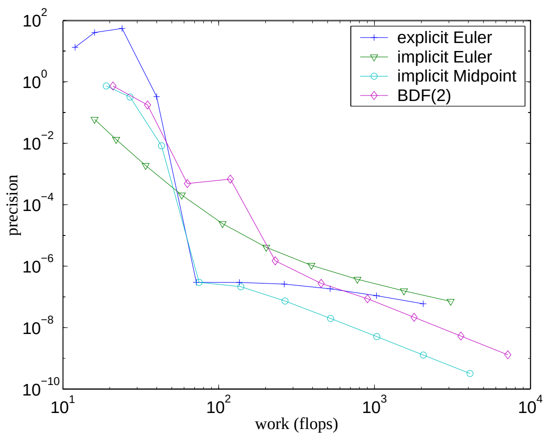
$$v_n - v_{n-1} = hf(x_n) \tag{57}$$

$$x_{n+1} - x_n = hv_n, \tag{58}$$

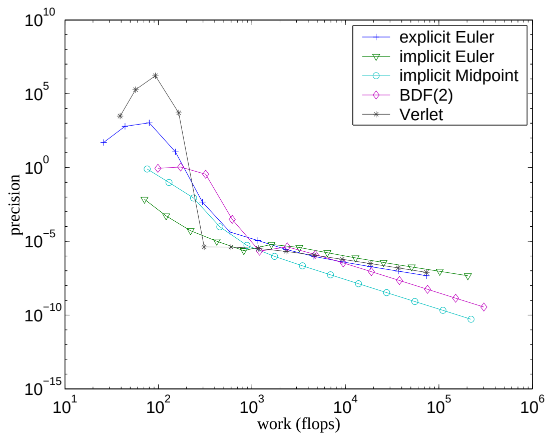
which omits the half steps and staggered grids from above. Now for second order equations which do not possess the form of (51) one may replace $f(x_n)$ by $f(x_n, v_{n-1})$ at the expense of some stability. Correctly the replacement had to



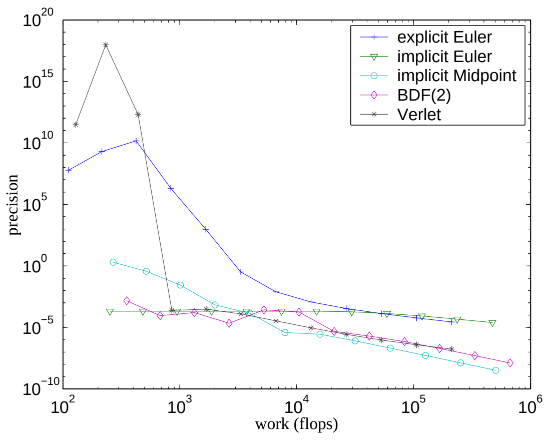
(a) Example 1a: $y' = 2y$



(b) Example 1b: $y' = -15y$



(c) Example 2: $y'' = -5y' - 5y$



(d) Example 3

Figure 11: *Work precision diagrams for the implicit Euler, implicit midpoint, BDF(2) and Verlet (whenever possible) methods. The results for Euler are for comparison and the same as in figure 9.*

be with $f(x_n, v_n)$ but this would result in a implicit method. Now we can apply the method to examples 2 and 3.

Figure 11 shows the work precision diagrams for the implicit methods. Even for large time steps these methods give approximations to the exact solution. After a somewhat uneven convergence phase all the explicit methods converge smoothly for decreasing time step to the exact solution with a slope given by their order.

Up to now we have omitted a discussion of the stability properties of the leapfrog method. From the examples it can be observed that the method cannot be unconditionally stable. Indeed some more delicate computations show that the method only delivers a bounded solution for arbitrary $h > 0$ for purely oscillatoric equations, i.e. for second order ordinary differential equations of the form (51) (with no damping term) and a contractive right hand side. Nevertheless the method remains well behaved in the presence of low damping. In figure 13 we applied the Verlet method to the undamped wave equation $y'' = 5y$ and it is clearly one of the best choices over a wide range of accuracy requirements.

3.1.6. Selecting an efficient method

Which method is best for a certain application? This question is nearly impossible to answer a priori. The only choice is to try a set of methods and to evaluate which one performs

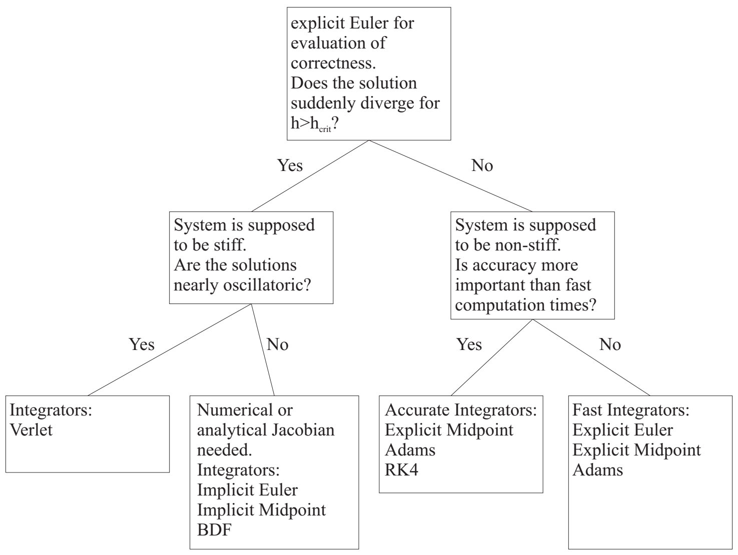

explicit Euler for evaluation of correctness. Does the solution suddenly diverge for h>h_crit?
Yes
No
System is supposed to be stiff. Are the solutions nearly oscillatoric?
System is supposed to be non-stiff. Is accuracy more important than fast computation times?
Yes
No
Yes
No
Integrators: Verlet
Numerical or analytical Jacobian needed. Integrators: Implicit Euler Implicit Midpoint BDF
Accurate Integrators: Explicit Midpoint Adams RK4
Fast Integrators: Explicit Euler Explicit Midpoint Adams

Figure 14: *Selecting a method.*

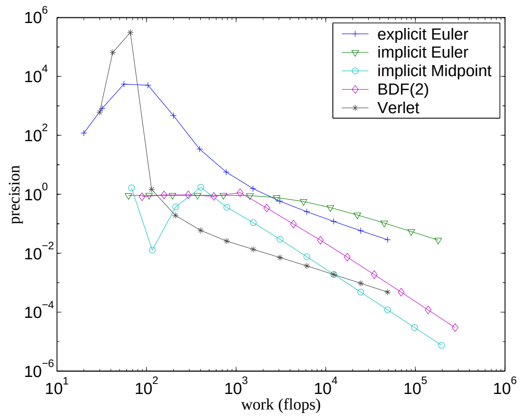


Figure 13: *Work precision diagram for the explicit/implicit Euler, implicit midpoint, BDF(2) and Verlet methods for the wave equation $y'' = -5y$ without damping.*

best. Choosing the methods to try, though can be done based on theoretical considerations and observations of the problem at hand. A possible strategy is shown in figure 14.

The same statement holds for predicting the efficiency of a method. Generally, implicit methods require more work per step. On the other hand one may be able to use time steps that are several magnitudes larger than the ones explicit methods would allow. Although accuracy will suffer, the integration won't be unstable. If evaluations of the right hand side function are cheap, a step with RK(4) is faster than an implicit step with BDF(4). On the other hand if it is cheap to compute a good sparse approximation to the jacobian, it may be more efficient to solve the linear system with a few cg iterations than to perform 4 full function evaluations.

3.2. Solving nonlinear systems

All implicit methods, unless they use a linear force formulation as the one described in section 2.2.6, require the solution of a nonlinear system. The implicit Euler method for example reduces our integration problem to the solution of the nonlinear system

$$Y_1 - hf(Y_1) - Y_0 = 0. \tag{59}$$

The other methods yield a system of similar form, namely

$$Y_1 - hf(\tfrac{1}{2}(Y_1 + Y_0)) - Y_0 = 0 \tag{60}$$

$$Y_1 - \frac{2}{3}f(Y_1) + \left(-\frac{4}{3}Y_0 + \frac{1}{3}Y_{-1}\right) = 0 \tag{61}$$

for the midpoint and BDF(2) rule, respectively. This is a nonlinear system of dimension $6N$. This system must be solved with Newton's method to allow for arbitrary step sizes independent of λ.

3.2.1. Newton's method

For the nonlinear system $G(Y) = 0$ we compute a numerical solution by the following algorithm:

Algorithm 1: Newton's Method

```
(1)  for $k = 1, 2, \ldots$ until convergence do
(2)      Compute $G(Y^{(k)})$.
(3)      Compute $J^{(k)} = \frac{\partial}{\partial Y} G(Y^{(k)})$.
(4)      Solve $J^{(k)} s^{(k)} = -G(Y^{(k)})$.
```

```
(5)     $Y^{(k+1)} := Y^{(k)} + s^{(k)}$
    end
```

Applying Newton's method reduces the problem to the successive solution of linear systems. In a classical Newton method this is achieved by Gaussian elimination. This introduces a lot of non-zero elements into the factors. Although reordering techniques alleviate the effects, this approach is currently too expensive for fast cloth animations.
A lot of authors[2, 39] use iterative methods to solve the linear system. We will also use the conjugate gradient (cg) method here to solve the linear systems in each Newton step. However, this changes the convergence behaviour of the outer Newton method, which is referred to as an inexact Newton method[35], given by algorithm 2.

Algorithm 2: Inexact Newton Method

```
(1) for $k = 1,2,\dots$ until convergence do
(2)     Compute $G(Y^{(k)})$.
(3)     Compute $J^{(k)} = \frac{\partial}{\partial y}(Y^{(k)})$.
(4)     Find $s^{(k)}$ with $J^{(k)}s^{(k)} = -G(Y^{(k)})) + r^{(k)}$,
        such that $\|r^{(k)}\| \leq \eta_k \|G(Y^{(k)})\|$.
(5)     $Y^{(k+1)} := Y^{(k)} + s^{(k)}$
    end
```

3.2.2. Residual control

The error of the iterative solution of the linear system is formulated in terms of the residual, which is easily computationally accessible, whereas the actual error cannot be computed. The tolerance of the linear iteration is decreased proportionally to the monotonically decreasing residual of the nonlinear iteration.
An analysis of this method[35] shows that it converges under rather weak additional assumptions. If the classical Newton method converges and the scalar tolerances $\eta^{(k)}$ are uniformly bounded by an $\eta < 1$, the inexact method converges. In literature the $\eta^{(k)}$ are referred to as *forcing terms*. Note that this additional assumption is also necessary: For $\eta = 1$, $s^{(k)} = 0$ would be admissible and the iteration would stagnate.
The inexact method then at least converges linearly, whereas Newton converges superlinearly. By choosing the η_k to converge to zero sufficiently fast[35], the convergence of the inexact Newton method can be forced to have an order > 1. In a neighbourhood of the solution the convergence usually speeds up. By extrapolating the solution of the previous time step we obtain a good initial value for the new solution and the method converges quickly using the constant bound $\eta = 0.02$ without imposing a too strict tolerance on the linear solver.

3.2.3. Inexact simplified Newton methods

The efficiency of the Newton method can be further improved by another approximation. In the simplified version of Newton's method the Jacobian $J^{(k)}$ is approximated by $J^{(0)}$. Such a scheme can be rewritten in the form of an inexact Newton method, if the linear system is written as follows and J is chosen as approximation to $J^{(k)}$

$$\begin{aligned} Js^{(k)} &= -G(Y^{(k)}) + (J - J^{(k)})s^{(k)} + r^{(k)} \\ &=: -G(Y^{(k)}) + \tilde{r}^{(k)} \end{aligned} \tag{62}$$

The residual $r^{(k)}$ is replaced by the larger $\tilde{r}^{(k)}$, which can be bounded if $J \approx J^{(k)}$. By choosing $\tilde{\eta}^{(k)}$ appropriately, the method still converges. In fact, we trade some accuracy approximating $J^{(k)}$ against accuracy in solving the linear system and up to a certain limit the method still behaves as before.

This degree of freedom can be further exploited by even not computing $J^{(0)}$ but a sparser approximation of it. In [24], cloth is modelled using the continuity based approach from section 2.2.5, and the choice of the approximated Jacobian is motivated by observing that the dominating stiffness is induced into the system by linear spring forces from (13). These are split into a linear and a nonlinear part.
Then, the right-hand side of the system is approximated by the linear expression

$$F_{\text{lin}}(x,v) = \sum_{j|(i,j)\in E} \left[\frac{k_{ij}}{l_{ij}^2}(x_i - x_j) + \frac{d_{ij}}{l_{ij}^2}(v_i - v_j) \right] \tag{63}$$

Writing this equation in matrix notation

$$F_{\text{lin}}(x,v) = Kx + Dv, \tag{64}$$

the full Jacobian of the system is approximated by the Jacobian of $F_{\text{lin}}(x,v)$. Then, (62) is applied with

$$J = I - h\gamma \begin{pmatrix} 0 & 0 \\ K & D \end{pmatrix}, \tag{65}$$

where h is the time step and γ depends on the integration method used. This system of dimension $6N$ can be reduced to a system of dimension $3N$ by exploiting the linear relation between position and velocity [23].
This choice of the Jacobian has two major advantages over the full Jacobian. First, J is inexpensive to compute and only changes when either the material constants or the step size changes. Second, we reduce the entries in the Jacobian to approximately a third of the entries in the sparsity pattern of the full Jacobian. Hence an iteration of the linear solver only requires a third of the original time. Obviously this is a major speed-up for the solver. The resulting algorithm is surprisingly simple.

Algorithm 3: Inexact Simplified Newton's Method

```
(1) Compute $J \approx \frac{\partial}{\partial Y} G(Y^{(0)})$.
(2) for $k = 1,2,\dots$ until convergence do
```

```
(3)    Compute $G(Y^{(k)})$.
(4)    Find $s^{(k)}$ with $Js^{(k)} = -G(Y^{(k)})) + r^{(k)}$,
         such that $\|r^{(k)}\| \leq \tilde{\eta}_k \|G(Y^{(k)})\|$.
(5)    Update $Y^{(k+1)} := Y^{(k)} + s^{(k)}$
    end
```

3.2.4. Adaptive time stepping

Newton's method can also be used to control the step size of the ODE solver. If the convergence of Newton's method is poor, the time step h is reduced such that the solution of the previous time step is a better start value for the current time step and achieves a faster convergence.

3.3. Comparison of methods

We are now able to describe many of the current approaches for cloth simulation in a unifying way.

Explicit integration methods, mostly together with a geometric post-correction step as described in section 2.1.1, have been used for instance by Provot [34], Volino [42], and Fuhrmann [18].

Baraff and Witkin [2] formulate nonlinear constraints and use their linear approximation for the construction of an implicit solution method commonly referred to as linear implicit Euler. This way the system to be solved also becomes linear and can be solved efficiently by a conjugate gradient method (see also [1]). This method corresponds to the solution of a nonlinear system with only one Newton iteration. Because the nonlinear part is not integrated, with high stiffness one may encounter problems which were addressed in [13, 24].

Following [2], implicit integration has been employed with varying physical models for instance in [39, 6, 5, 17]. Desbrun et al. [11, 32] also use a linear implicit method combined with Provots model [34] (section 2.1.1). But instead of linearizing the whole system, they split it in a linear and nonlinear part and use a precomputed inverse of A for solving the linear part of the equations. They don't aim at solving the equation completely, as they don't integrate the nonlinear term explicitly. Instead the angular momentum is corrected to account for the nonlinear part. With this algorithm one can neither change the stepsize h nor deal with an Jacobian depending on t. Debunne et al. [10] use a simplified version of a linear implicit Euler method. To solve the differential equation arising from their geometric nonlinear Green model, they apply an implicit Euler method using a single Newton iteration. The Jacobian is approximated by the 3x3 block-diagonal of the linear Cauchy tensor, allowing a very fast inversion and application.

4. Conclusions

We described physical models for cloth ranging from mass-spring and particle systems to finite element models. Moreover, we discussed and compared several explicit and implicit time integration methods. All these techniques form the basis of current physically based cloth simulation engines, and are suitable to be combined with hybrid approaches for real-time animation of clothing such as [8, 9].

References

[1] U. Ascher and E. Boxerman. On the modified conjugate gradient method in cloth simulation. *The Visual Computer*, 2003.

[2] David Baraff and Andrew Witkin. Large Steps in Cloth Simulation. In *Computer Graphics (Proc. SIGGRAPH)*, pages 43–54, 1998.

[3] Javier Bonet and Richard D. Wood. *Nonlinear continuum mechanics for finite element analysis*. Cambridge University Press, 2000.

[4] Dieter Braess. *Finite Elemente*. Springer, 1997.

[5] K.-J. Choi and H.-S. Ko. Extending the Immediate Buckling Model to Triangular Meshes for Simulating Complex Clothes. In *Eurographics Short Presentations*, pages 187–192, 2003.

[6] Kwang-Jin Choi and Hyeong-Seok Ko. Stable but Responsive Cloth. In *Computer Graphics (Proc. SIGGRAPH)*, pages 604–611, 2002.

[7] Philippe G. Ciarlet. *Mathematical Elasticity. Vol. I.* North-Holland Publishing Co., Amsterdam, 1992.

[8] F. Cordier and N. Magnenat-Thalmann. Real-time Animation of Dressed Virtual Humans. *Computer Graphics Forum*, 2002.

[9] F. Cordier, H. Seo, and N. Magnenat-Thalmann. Made-to-Measure Technologies for an Online Clothing Store. *IEEE Computer Graphics and Applications*, 23(1):38–48, 2003.

[10] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. Dynamic Real-Time Deformations using Space and Time Adaptive Sampling. In *Computer Graphics (SIGGRAPH 01 Proceedings)*, 2001.

[11] Mathieu Desbrun, Peter Schröder, and Alan Barr. Interactive Animation of Structured Deformable Objects. In *Graphics Interface*, pages 1–8, June 1999.

[12] Manfredo P. DoCarmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1976.

[13] Bernhard Eberhardt, Olaf Etzmuß, and Michael Hauth. Implicit-Explicit Schemes for Fast Animation with Particle Systems. In *Eurographics Computer Animation and Simulation Workshop*, 2000.

[14] Bernhard Eberhardt, Andreas Weber, and Wolfgang

Straßer. A Fast, Flexible Particle-System Model for Cloth Draping. *IEEE Computer Graphics and Applications*, 16(5):52–59, 1996.

[15] Jeffrey W. Eischen, Shigan Deng, and Timothy G. Clapp. Finite-Element Modeling and Control of Flexible Fabric Parts. *IEEE Computer Graphics and Applications*, 16(5):71–80, 1996.

[16] Olaf Etzmuß, Joachim Groß, and Wolfgang Straßer. Deriving a Particle System from Continuum Mechanics for the Animation of Deformable Objects. *IEEE Transactions on Visualization and Computer Graphics*, 2003.

[17] Olaf Etzmuß, Michael Keckeisen, and Wolfgang Straßer. A Fast Finite Element Solution for Cloth Modelling. *Proc. Pacific Graphics*, 2003.

[18] A. Fuhrmann, C. Groß, and V. Luckas. Interactive Animation of Cloth including Self Collision Detection. In *Journal of WSCG*, 2003.

[19] Eitan Grinspun, Petr Krysl, and Peter Schröder. CHARMS: a simple framework for adaptive simulation. In *Computer Graphics (Proc. SIGGRAPH)*, 2002.

[20] Joachim Groß, Olaf Etzmuß, Michael Hauth, and Gerhard Bueß. Modelling viscoelasticity in soft tissues. In *Int. Workshop on Deformable Modeling and Soft Tissue Simulation*, 2001.

[21] E. Hairer and G. Wanner. Solving Ordinary Differential Equations I & II. Springer-Verlag, Berlin, 1996.

[22] E. Hairer and G. Wanner. Solving Ordinary Differential Equations II. Springer-Verlag, Berlin, 1996.

[23] M. Hauth, O. Etzmuß, and W. Straßer. Analysis of Numerical Methods for the Simulation of Deformable Models. *The Visual Computer*, 2003.

[24] Michael Hauth and Olaf Etzmuß. A High Performance Solver for the Animation of Deformable Objects using Advanced Numerical Methods. In *Proc. Eurographics*, 2001.

[25] Michael Hauth and Wolfgang Strasser. Corotational simulation of deformable solids. In *Proc. WSCG 2004*, pages 137–145, 2004.

[26] Donald H. House and David E. Breen, editors. *Cloth Modeling and Animation*. A K Peters, 2000.

[27] Lukas Janski and Volker Ulbricht. Numerical simulation of mechanical behaviour of textile surfaces. *Zeitschrift für Angewandte Mathematik und Mechanik*, 80(S2):S525–S526, 2000.

[28] Young-Min Kang, Jeong-Hyeon Choi, Hwan-Gue Cho, Do-Hoon Lee, and Chan-Jong Park. Real-time Animation Technique for Flexible and Thin Objects. In *WSCG*, pages 322–329, February 2000.

[29] S. Kawabata. *The Standardization and Analysis of Hand Evaluation*. The Textile Machinery Society of Japan, Osaka, 1980.

[30] E. Klingbeil. *Tensorrechnung für Ingenieure*. BI Wissenschaftsverlag, 1989.

[31] L. D. Landau and E. M. Lifschitz. *Elastizitätstheorie*. Akademischer Verlag, 1989.

[32] M. Meyer, G. Debunne, M. Desbrun, and A. Barr. Interactive Animation of Cloth-like Objects in Virtual Reality. *The Journal of Visualization and Computer Animation*, 12(1):1–12, 2001.

[33] M. Müller, J. Dorsey, L. McMillan, R. Jagnow, and B. Cutler. Stable Real-Time Deformations. In *Proc. of SIGGRAPH Symposium on Computer Animation 2002*, 2002.

[34] Xavier Provot. Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior. In *Graphics Interface '95*, pages 147–154, 1995.

[35] W. C. Rheinboldt. *Methods for Solving Systems of Nonlinear Equations*, volume 70 of *CBMS-NSF regional conference series in applied mathematics*. SIAM, second edition, 1998.

[36] Hans Stephani and Gerhard Kluge. *Theoretische Mechanik*. Spektrum Akademischer Verlag, 1995.

[37] D. Terzopoulos and K. Fleischer. Deformable models. *The Visual Computer*, 4:306–331, 1988.

[38] P. Volino and N. Magnenat-Thalmann. Developing simulation techniques for an interactive clothing system. In *International Conference on Virtual Systems and Multimedia '97*, pages 109–118, 1997.

[39] P. Volino and N. Magnenat-Thalmann. Implementing fast Cloth Simulation with Collision Response. In *Computer Graphics International Proceedings*, 2000.

[40] P. Volino and N. Magnenat-Thalmann. *Virtual Clothing*. Springer, 2000.

[41] P. Volino and N. Magnenat-Thalmann. Comparing Efficiency of Integration Methods for Cloth Animation. In *Computer Graphics International Proceedings*, 2001.

[42] Pascal Volino, Martin Courshesnes, and Nadia Magnenat-Thalmann. Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects. In *Computer Graphics (Proc. SIGGRAPH)*, pages 137–144, 1995.

Collision Detection

Pascal Volino and Nadia Magnenat-Thalmann
MIRALab, University of Geneva - CH-1211 Geneva, Switzerland

Abstract
Deformable objects cover a large range of applications in computer graphics and simulation, ranking from modeling techniques of curved shapes to mechanical simulation of cloth or soft volumes. Efficient collision detection is used in all these applications for ensuring consistent design and simulation.

1. Collision Detection Strategies

Unlike rigid bodies, deformable objects have evolving shapes. In most cases, this implies that their surface are curved. Adequate modeling techniques are needed to describe these objects. Among the most popular, polygonal meshes and implicit surfaces (for example metaballs) are used. Whereas surfaces are usually described as polygonal meshes (usually flat polygons such as triangles or quadrangles, but possibly also curved patches such as Bezier patches or subdivision surfaces), volumes are most of the time also described by their bounding surfaces. Collision detection is usually performed on these surfaces.

The usual complexity of collision detection processes result from the large number of primitives that describe these surfaces. Most of collision detection applications need to compute which polygons of large meshes do actually collide. In most of the cases also, these meshes are animated (through user interaction or simulation processes) and collision detection has to be involved at each steps of these animations for ensuring immediate and continuous feedback to the animation control.

This creates a particular context that collision detection has to take advantage for optimal performance.

1.1. Hierarchy Structure

Most of efficient collision detection algorithms take advantage of a hierarchical decomposition of the complex scheme. This allows to avoid the quadratic time of testing extensively collisions between all possible couples of primitives.

There are two major ways of constructing such hierarchies:

* Space subdivision schemes, where the space is divided in a hierarchical structure. These are typically octree methods. Using such structure, a reduced number geographical neighbors of a given primitive are found in **log(n)** time (the depth of a hierarchy separating geographically **n** primitives) and tested for collisions against it.

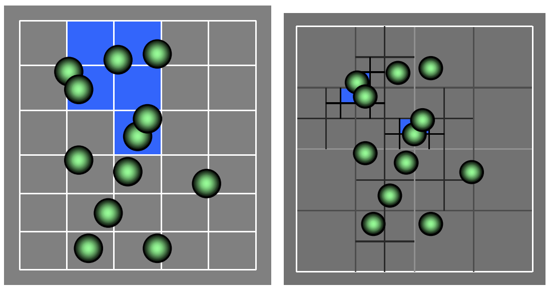


***Figure**: Space subdivision methods, flat (left) or hierarchical (right) rely on a partition of space into regions containing primitives.*

* Object subdivision schemes, where the primitives of the object are grouped in a hierarchical structure. These are typically methods based on bounding volume hierarchies. Using such structure, large bunches of primitives may be discarded in **log(n)** time (the depth of a well-constructed hierarchy tree of **n** primitives) through simple techniques such as bounding-volume evaluations.

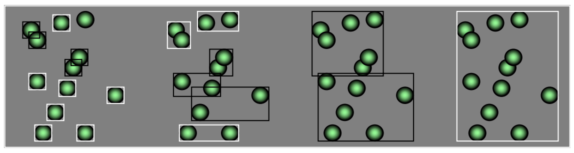


***Figure**: Object subdivision methods rely on a subdivision of the scene into groups of primitives.*

Other methods, such as voxel methods, projection methods or render overlap methods, usually belong to non-hierarchical space subdivision schemes.

Any of these methods may be used in the context of deformable objects. However, maximum efficiency is obtained by taking advantage of all invariants that could reduce the amount of time spent in computation. In the context of deformable surfaces which use extensively discretized surface animations (polygonal meshes or patches animated through motion of their control points), the major invariant to take advantage of is *the local invariance of the mesh topology*.

Unlike polygon soups, where the primitives are totally independent one from another, the primitives of a polygonal mesh maintain a constant adjacency structure which defines, in a local state, some constant geographic properties between these primitives. Hence, adjacent elements of a polygonal mesh have similar positions whatever the motion of the surface, during all the animation.

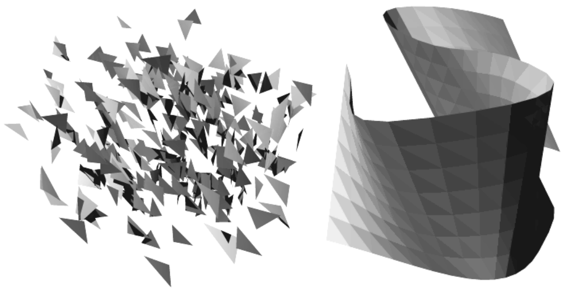


***Figure**: A polygon soup, and a polygonal mesh which maintains local position constancy.*

Considering that the topology of the mesh defines a constant native geographical neighborhood of the primitives of the mesh (which are usually non-colliding primitives), a good idea is to define the collision detection hierarchy based on this criteria. On the other hand, any topologically unrelated primitives ("far away" from each other on the mesh topology) which come to be geographically close to each other are very likely to be colliding.

This is why object-based hierarchies are very likely to be the best paradigm for defining an efficient collision detection algorithm for animated meshes. Compared to space subdivision methods, the main benefits are the following:

* In most cases, they can work on a constant hierarchy structure, which does not need to be updated along the animation.

* The topology of the hierarchy reflects the adjacency of the surface regions described in it. Adjacency information may thus be used in various optimizations, such as the approach for optimizing self-collision detection, as discussed later.

2. Bounding Volume Hierarchies

Hierarchical collision detection heavily relies on bounding volumes. Performance depends on:

* How tight they enclose the object part corresponding to their hierarchy level, for avoiding at best false positives in the collision detection tests.

* How efficiently they can be computed from a primitive.

* How efficiently they can be combined, for propagation up in the hierarchy.

* How efficiently they can be updated if the object is animated.

* How efficiently a collision test can be performed between two volumes.

The adequate bounding volume is chosen so as to offer the best compromise between these factors, and this is quite dependent on the simulation context (kind and number of object primitives, kind of animation...).

Among popular choices,

* Bounding spheres or ellipsoids.

* Bounding boxes or Discrete Orientation Polytopes.

2.1. Choosing the Suitable Bounding Volume Scheme

The major choice is to be set between three types of bounding volumes:

* Axis-independent volumes, such as spheres.

* Axis-aligned volumes which are defined in absolute world coordinates.

* Object-oriented volumes which are defined in local object coordinates.

This choice is particularly important when working with animated objects. In that context, the most important issue is to reduce the time taken for updating the bounding volume hierarchy for each step of the animation.

2.1.1. Object-Oriented Volumes

The first idea is to attempt to skip this recomputation whenever possible. This is possible when working with rigid objects, which only have rigid motion in the scene (translation and rotation). In this case, rather than defining the volumes in world coordinates, attaching them to object coordinates removes the need of updating them.

Another motivation for using axis-oriented volume is the optimization of the bounding tightness. Hence, is the object is flat or elongated, there are large benefits in using an adequately aligned volume that fits the shape tightly.

The difficulty is however moved to another place in the collision detection process: Combining and comparing bounding volumes defined in different coordinate systems. While the combination process is usually performed once for all in the hierarchy describing rigid objects, testing for collisions between bounding volumes require coordinate transformation computations that may take significant computation resources. Some schemes also expand the volumes in order to compensate the change of axis rotation, at the expense of bounding tightness. This difficulty can also be avoided by using axis-independent volumes (spheres), but these are very inefficient in bounding tightness (specially for flat or elongated objects).

2.1.2. Axis-Aligned Volumes

The other idea is to make perform extensively the recomputation of the bounding volume hierarchy at each position change, with operations on bounding volumes made as simple and efficient as possible. This is actually the best approach for deformable objects, as there is no way to efficiently exploit shape invariance.

The most popular choice for this are axis-aligned bounding boxes. They are essentially defined by the minimum and maximum coordinates of the enclosed objects, which can be computed very easily. Combining bounding boxes is also trivial, and collision test between two boxes is simply evaluated by testing min-max overlap along all coordinates.

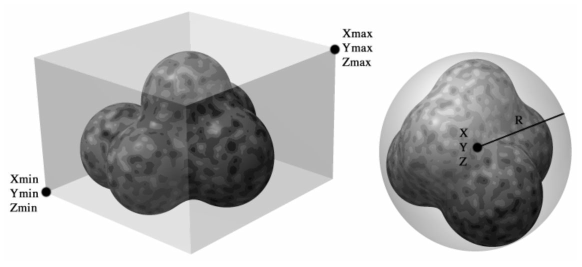


Figure*: A bounding box and a bounding sphere.*

One of the biggest issues with axis-aligned bounding boxes is that they do not always offer good bounding tightness, particularly when objects are flat or elongated along a diagonal direction. Of course, reverting to their object-oriented variant would void the benefits of fast collision tests. The other solution is to use a generalization of bounding boxes along more directions than the native axes alone.

2.2. Beyond Bounding Boxes: Discrete Orientation Polytopes

Bounding boxes are based on min-max interval computations on the directions defined by the natural world coordinate axes. While this is sufficient for defining bounding volumes, why not adding more directions? It's like "cutting away" a bounding volume along particular directions so as to obtain tighter bounding volumes.

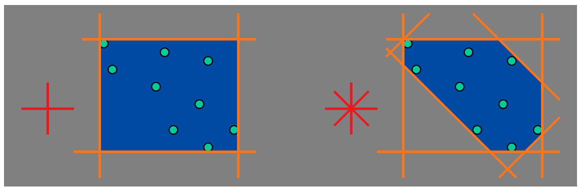
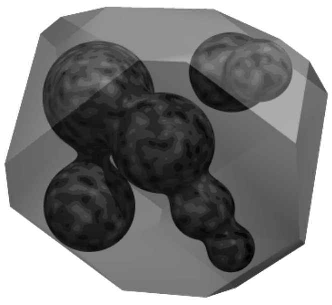


Figure*: Discrete Orientation Polytopes are generalizations of bounding boxes along arbitrary directions.*



Mathematically, given a set of directions **Di**, a Discrete Orientation Polytope (DOP) a set of vertices **Vk** is defined by two vectors **Mi** and **Ni** such as:

$\mathbf{Mi} = \min_k(\mathbf{Di} \cdot \mathbf{Vk})$ and $\mathbf{Ni} = \max_k(\mathbf{Di} \cdot \mathbf{Vk})$

The union of two DOPs is computed as follows:

$\mathbf{Mi} = \min(\mathbf{Mi}_1, \mathbf{Mi}_2)$ and $\mathbf{Ni} = \max(\mathbf{Ni}_1, \mathbf{Ni}_2)$

Two DOPs do not intersect if the following condition is met for at least one value of **i**:

$\mathbf{Mi}_1 > \mathbf{Ni}_2$ or $\mathbf{Mi}_2 > \mathbf{Ni}_1$

The adequate set of directions to be considered should describe a sampling of the direction space as uniform as possible. Of course, this sampling is point-symmetric, since a direction vector also represents its opposite direction.

In 3D, the easiest approach is to construct a set of directions starting from the cube (standard bounding box, which is a DOP of 6 directions):

D0=(1,0,0) **D1**=(0,1,0) **D3**=(0,0,1)

... and add the cube diagonals (directions to its vertices):

D4=(√3,√3,√3) **D5**=(√3,-√3,-√3) **D6**=(-√3,√3,-√3) **D7**=(-√3,-√3,√3)

... for obtaining a DOP describing 14 directions. 12 additional directions can be added by using the square diagonals (directions to its edge centers):

D8=(0,√2,√2) **D9**=(0,√2,-√2) **D8**=(√2,0,√2) **D9**=(-√2,0,√2) **D10**=(√2,√2,0) **D11**=(√2,-√2,0)

Note that for this set, it is not actually necessary to normalize the direction vectors, relieving the necessity of performing multiplications for computing a DOP enclosing a set of vertices. However, normalized direction vectors offer good evaluation of the size of the DOP through the min-max interval width in each direction, as well as a simple expansion scheme for distance-based collision detection.

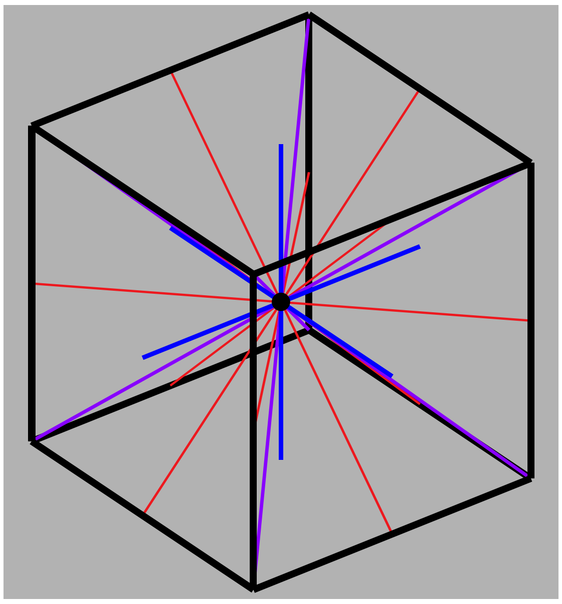


Figure*: The 26 direction set defined by a cube.*

When more directions are needed, it is also possible to construct a set from a dodecahedron (12 directions), adding to it directions to its vertices (20 additional directions) and to its edge centers (30 additional directions). The benefit of this set is a better distribution of directions. However, multiplications cannot be avoided for computing the DOP of a set of vertices.

Ultimately, with a huge number of directions, DOP tend to get the shape of the convex hull of the enclosed objects.

The choice to be made is actually to find the optimal number of directions defining the DOP. In one hand, the more directions, the tighter the DOP fits to the convex hull of the object (or to the object itself if it is convex). In the other hand, the more directions, the more computation is needed for computing the DOPs and evaluating their intersections. The best choice is actually dependent on the shape of the objects (flat or elongated objects will get more benefits from tight DOPs than round or concave ones), and also the extra cost of performing explicit collision detection on object marked as colliding by rough quickly-evaluated DOPs (for instance, it takes much more time evaluating collisions between curved patches than flat polygons). The only way of finding the best compromise is to carry out experimental benchmarks on various examples typical of the wanted simulation context.

3. Collision Detection on Polygonal Meshes

Polygonal meshes are the most popular way of describing deformable objects. They may either represent the surface object itself (for example, cloth), or the boundary of a volume object. Collision detection is usually carried out by finding which of the polygons of the meshes are actually colliding.

Figure: *Objects animated and simulated as polygonal meshes.*

Given the considerations discussed above, the typical best choices for detecting collisions on animated polygonal meshes are the follows:

* Use of a hierarchical bounding volume scheme constructed on the objects.

* Use of efficient axis-aligned bounding volumes, such as DOPs.

Choosing a constant hierarchy built on the mesh seems to be quite a good assumption, as for typical objects, the distance on the mesh is quite a good evaluation of the native distance that should separate features of the mesh, and bounding volumes enclosing a small region of the mesh are likely to remain small for any usual deformation.

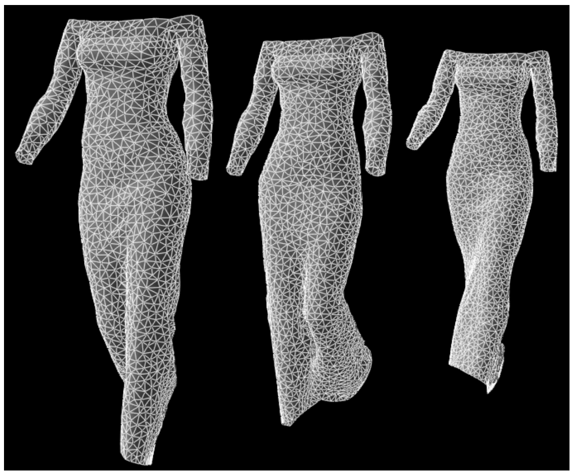


Figure: *Moving polygonal meshes describing the cloth surface.*

1.2. Building Mesh Hierarchies

The performance of the collision detection algorithm is highly dependent on how well-constructed the hierarchy tree is.

The tree should be well conditioned, meaning:

* Each node should have at maximum O(1) children.

* The tree should be balanced, so the tree should have O(log n) maximum depth.

Another essential quality of the tree is to offer minimal bounding volumes for each tree node. This means that the surface elements represented by a tree node should have maximum vinicity. Thus, it is important to build the hierarchy consistently to the topology of the mesh, which is a good evaluation on the relative positioning of the mesh elements.

One efficient solution is to build the hierarchy tree using an ascending process. First the leaf nodes of the tree are constructed, corresponding to all the individual polygons of the surface. Then, an upper layer of the hierarchy is built by grouping two or three nodes in a common parent node. The tree is built level by level until only one element remains, which is the root node of the tree.

The grouping can be performed by a region-merging algorithm. Initially, each surface polygon is assigned a unique region ID, which identifies a group of polygons. During the grouping process, candidate edges that separate two different groups (two polygons having different ID) are considered. One of these edges is then selected, and all the polygons of one group (usually the smallest) are assigned the ID of the polygons of the other group. This algorithm generates groups of connected regions, within each of which all the polygons are connected by at least one edge. A counter should also be included in each group, to keep track of the number of subgroups that have been merged in the group. It can be used to limit the number children a group has. This algorithm is in fact very close to automatic labyrinth-generation algorithms which are based on the same region-merging scheme.

It is important to determine efficiently which nodes to group in order to create the parent node. First, only adjacent surface regions should be connected. Secondly, the regions generated should be as “well shaped” as possible, i.e. closer to a disk than a elongated or sprawling branched structure. The resulting bounding boxes will therefore be as compact as possible, whatever the result of any reasonable 3D deformation of the surface.

A good way to characterize “well-shaped” polygons is to compare its contour length to its surface area. The algorithm can construct groups by maximizing the “shape factor” **sqrt(SurfaceArea) / ContourLength**. At a given hierarchy level, this ratio is computed for any group that can be generated by the removal of an edge. The potential groups are then sorted by this criteria, and new groups are then constructed wherever possible, according to this sorting. Between two hierarchy levels, the algorithm first tries to merge groups into pairs, and then merges the remaining groups into these new groups.

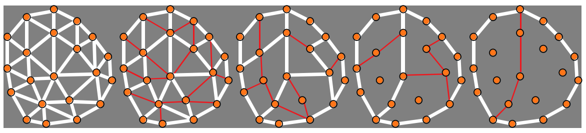


***Figure**: Building a mesh hierarchy*

Though this proposed grouping process does not necessarily yield hierarchy regions that globally optimize the shape factors, the results obtained are however quite acceptable for our application. More precisely, the self-collision detection algorithm remains efficient as long as the bounding volumes of non-adjacent hierarchy groups (that do not share at least a common vertex) do not intersect. Using the proposed algorithm, this feature is verified almost everywhere in usual polygonal meshes.

This algorithm should be implemented to build a hierarchy on any polygonal mesh that will be involved in collision detection. This computation should only be performed as preprocessing, and does not have to be performed for each collision detection when the surfaces have moved.

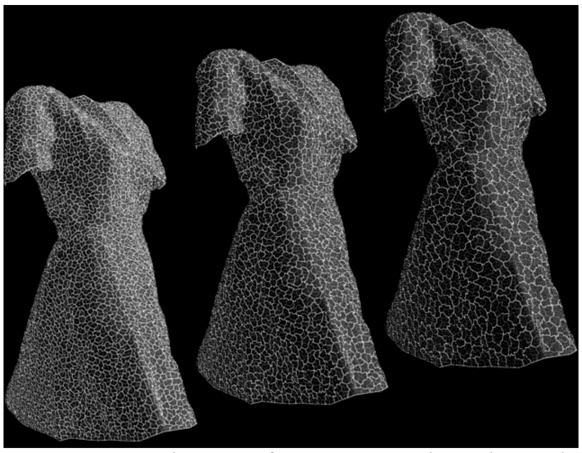


***Figure**: Hierarchisation of a 50 000 triangle mesh: Levels 5 to 7.*

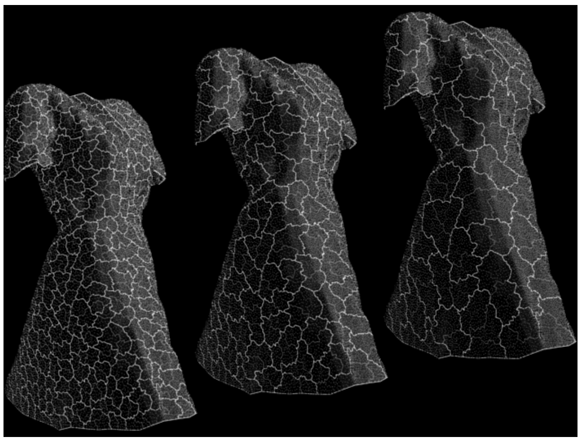


***Figure**: Hierarchisation of a 50 000 triangle mesh: Levels 8 to 10.*

3. Self-Collision Detection

There are very few algorithmic differences between self-collision detection within one object and collision detection between two separate objects. Ultimately, a hierarchical algorithm will end up splitting a single object into separate pieces, and perform usual collision detection between these pieces.

There is actually a major performance issue related to self-collision detection, related to adjacent elements actually seen as "colliding" by usual bounding-volume tests.

3.1. Why Self-Collision Detection is so Inefficient

Self-collision detection pertains to collisions between elements of the same surface. Of course, neighboring elements of the same surface are naturally in contact to each other. Any collision detection algorithm is designed to detect geometric contact between elements, and thus will be misled by these adjacent elements, and will consider them as colliding elements.

The number of adjacencies is usually proportional to the total number of elements in one surface. In a triangular mesh, the adjacency number is roughly 1.5 times the total number of triangles for common-edge adjacencies and 6 times the total number of triangles for common-vertex adjacencies.

The time spent detecting collisions is proportional to the number of colliding elements multiplied by the logarithm of the total number of elements. Typically, the number of self-colliding elements is very small compared to the total number of elements and often null if no collisions occur. Thus, “detecting” all the adjacencies as if they were collisions is a great waste of time, particularly in situations with very few collisions.

For efficiency, an algorithm should be designed to ignore all collisions that in fact are only element adjacencies.

3.2. Optimizing Self-Collision Detection with Curvature Evaluation

The curvature criteria is expressed as follows:

A flat surface cannot have self-collisions. On the other hand, if a surface has self-collisions, it must be bent enough to form a loop.

Is there a way to formalize this intuition? Curvature appears to be the key for achieving efficiency in self-collision detection.

When there are self-collisions on a surface, at any geometrical intersection point, the surface appears twice. For obtaining such configuration, the surface has to be bent enough to form a loop.

***Figure**: Self-collisions only occur if the surface is bent enough for creating a loop.*

This condition cannot be met if there exists a direction for which the orthogonal projection of the surface does not exhibit folds. If the surface is continuous, this means that all the normals of the surface have a dot product of constant sign with that direction vector.

We also have to consider self-collisions that may occur because of the shape of the surface boundary. This may also exhibit loops even if the surface is almost flat, when the surface contour itself exhibits a loop causing self-intersection. An additional test has therefore to be done on the surface contour. Having found a projection direction for the surface curvature, a sufficient criteria for non-intersection is that the projection of the surface contour along this direction should not self-intersect.

Thus, a surface does not have self-collisions if the following criteria are met:

* Let **S** be a continuous surface in Euclidean space, delimited by one contour **C**.

if

- There exists a vector **V** for which **N.V > 0** at (almost) every point of **S** (**N** being the normal vector of the surface at the considered point)

and

- The projection of **C** on a plane orthogonal to **V** along the direction of **V** has no self-intersections

then

- There are no self-collisions on the surface **S**.

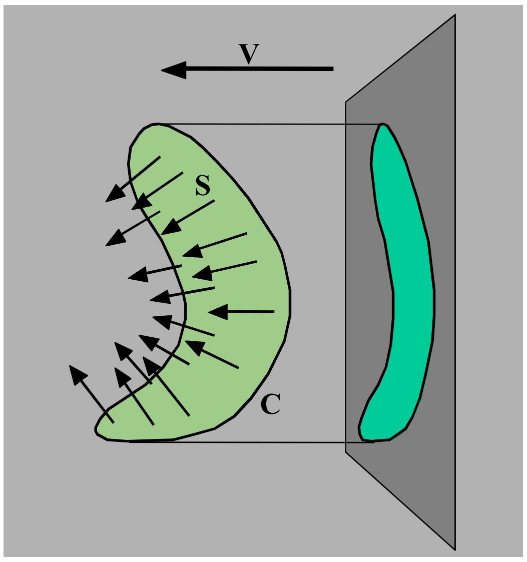


***Figure**: The curvature criteria.*

Such criteria allow us to efficiently discard, from the detection process, "almost flat" surface regions that will not exhibit internal self-collisions. The union of two adjacent surface regions may also be "almost flat" so we need not detect collisions between these two regions. In particular, adjacent elements need not be checked for collisions. Implemented efficiently, this criterion will allow us to deal with the major cause of inefficiency of self-collision detection.

It is now important to adapt the hierarchical collision detection scheme to include this criteria. As discussed, the general hierarchical collision detection algorithm works with bounding boxes. There are two main processes:

* For detecting collisions within a surface region, collisions are detected within and between all the children of its corresponding hierarchy tree node.

* For detecting collisions between two surface regions, collisions are detected between the respective children nodes.

Collisions between two nodes may be efficiently detected using a bounding-box evaluation: if the bounding boxes do not intersect, there are no intersections. However, in the standard hierarchical algorithm, there are no bounding box techniques for self-collision detection within one node. Replacing the bounding box test by a curvature test can overcome this limitation.

For the self-collision stage, how to integrate curvature considerations is clear: Collisions should be detected within one node only if the corresponding surface does not match the curvature criteria.

When detecting collisions between two nodes, we should consider two cases: Dealing with two adjacent surfaces, collisions should be detected between the nodes only if the corresponding surface union does not verify the curvature criteria (obviously, the bounding boxes will always intersect, so the curvature criteria acts as a good replacement test). Dealing with non adjacent surfaces, the standard bounding box criteria should be used.

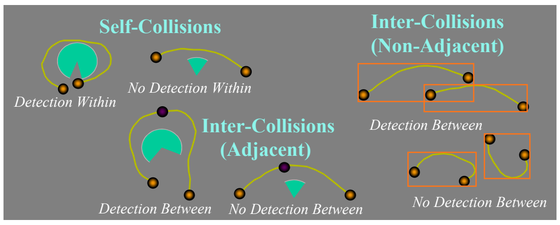


***Figure**: Using curvature or bounding volumes: The different cases.*

Thus, the curvature criteria is incorporated into the hierarchical algorithm by replacing the bounding volume test by a curvature test within surface regions or between adjacent surface regions.

Given a curved surface, we need to determine the existence of, and find, a compatible direction vector that has positive dot product with all normal vectors of the surface. In our discrete model of the surface, that means that this vector should have positive dot product with the normals of all the surface polygons.

Similar to what was done for the bounding boxes, “direction boxes” that contain the allowable directions will be propagated up to the hierarchical parents.

Assimilating the direction space to a sphere, the allowable direction box of a flat mesh polygon is a half sphere. When we consider two polygons, the globally allowable direction is the common part of the corresponding two sphere halves. This gives us the mechanism that should be propagated upwards in the tree hierarchy. The resulting direction box at the root of the tree may be empty, and in this case no suitable direction can be found for the whole surface. If not empty, any vector within the direction box is suitable.

Unlike volumes boxes that can be represented efficiently using space coordinates, there is no easy way to describe our direction boxes exactly. One solution is to use "direction cones" defined by a direction and an aperture angle. The bounding tightness is however very limited and combining two cones is not an efficient operation.

That difficulty can also be addressed by building a discrete set of direction vectors that will represent our direction space. The same set of sample directions can be used as the ones used for defining the DOPs used as bounding volumes: Using the 26 directions toward the face centers, vertices and edge centers of a cube, the angle accuracy is around 25°. The more directions used, the more accurate are the direction evaluations, at the expanse of computation time.

During the update process of the bounding volumes in the hierarchy tree, the direction boxes are updated as well. For each polygon of the mesh is computed the set of direction from the sample set that have positive positive dot product with the normal of the polygon. The result is stored in an array of boolean. Propagation up in the tree is done by combining these arrays with element-wise boolean AND operations. Then for any collision test, the low curvature criteria is met if TRUE elements remains in the array, and the corresponding directions are the directions that meet the dot product requirements for all the corresponding surface region.

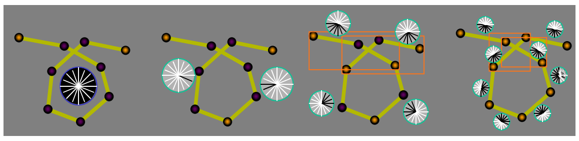


***Figure**: Combining curvature boxes and bounding volumes during the detection process.*

Another major problem is the adjacency test: Given two arbitrary nodes in the hierarchy, are the corresponding surfaces adjacent? (Do they have at least one common vertex?)

This represents the major difficulty of our algorithm. This adjacency should be detected efficiently (the computation complexity should be O(log n)), and the extra storage in the tree data structure should be reasonable (the extra information for each node should be O(1)).

For each hierarchical node, a list of vertices should be defined, these being the vertices surrounding the corresponding surface region. Several circular lists should be considered when dealing with non-connected surfaces or surfaces having holes. Obviously, if two nodes are adjacent, they have at least one of these vertices in common.

Not all the boundary vertices should be stored, but only the vertices that separate two different surface regions among those of the highest hierarchy level that also do not include the region being considered. “Outside” is considered as a particular surface region. Whatever the node level and the total number of polygons surrounding this surface region, the number of stored vertices is approximately constant, and for usual meshes hardly exceeds six. As this adjacency information only depends on the mesh topology, this stage is usually performed once in the preprocessing stage.

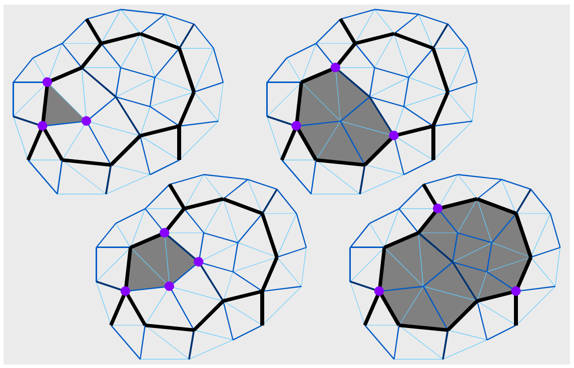


***Figure**: Storing region boundary vertices in the hierarchy tree: The number remains roughly constant whatever the level in the hierarchy.*

Adjacency testing is then performed easily: Two hierarchy nodes are adjacent if and only if they have at least one common vertex among those stored for these two nodes. This test is performed in O(1).

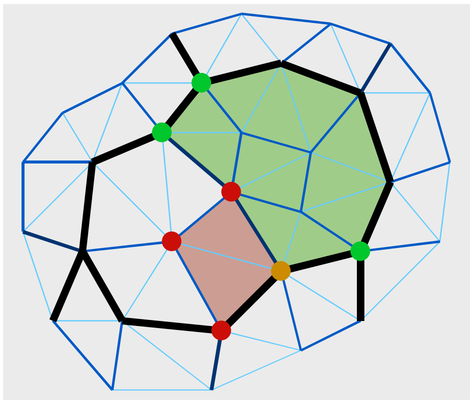


***Figure**: Two adjacent (point-connected) surface regions of the hierarchy have at least one common stored vertex.*

The boundary shape of the surface may also be the cause of self-collisions. In most cases, this happens when an elongated strip is fold so as to produce a cone-shaped surface. Such collisions are also very likely to happen around sharp concavities of the surface contour, where minimal folds can also produce self-collisions.

The most systematic solution would be to construct, for any surface region fulfilling the surface normal criteria, the 2D projection of the surface boundary on a plane orthogonal to one of the found directions. Despite flatness, collision detection should be carried out within or between children containing boundary intersections. This 2D collision detection process can also be based on a similar kind of curvature optimization. This would however require a significant amount of extra computation, along the data required for managing a contour-based hierarchies.

Practical test have however shown that these tests are rarely significant in most "real-world" situations involving for example garment simulation or deformation of soft objects. These tests may however improve detection of long objects with large curvature deformations (simulation of long ribbons) or surfaces with complex non-convex contours (cuts, holes).

These marginal situations may be addressed using various low-cost approximate techniques. The simplest method is to "expand" the direction boxes with a certain angle for mesh elements that are located on the boundary of the surface. This angle may also be increased for elements adjacent to non-convex boundary locations. The larger the angle, the most systematic the collision detection is, at the expanse of computation time.

3.3. Efficiency

The main interest of this algorithm is its efficiency in detecting self-collisions: Hierarchy regions that are not curved enough to contain self-collisions are efficiently omitted from the detection process. The following figure shows the regions considered when performing collision detection between two objects, as well as self-collision detection within these objects also. As a result, the algorithm efficiently focuses on the intersecting parts of the surfaces.

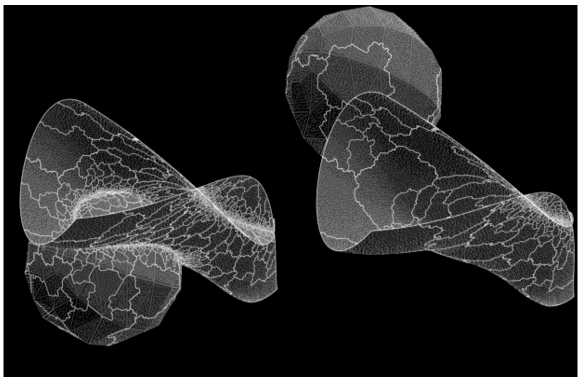


***Figure**: Collision detection is focused on the colliding parts of the surface.*

The execution time required for performing collision detection is subdivided as follows:

* Update of the bounding volumes of the hierarchy tree.

* Update of the direction boxes of the hierarchy tree, if self-collision detection is required.

* Running the collision detection algorithm on the hierarchy tree.

The time required for the two first steps is proportional to the number of mesh elements. The involved computations are quite reduced for mesh elements (evaluation of min-max of linear combination of vertex coordinates and normal computation of polygons usually also required for other purposes (mechanics, rendering...), and dot product with a set of directions). Their propagation along the hierarchy tree is also trivial (min-max interval merges, boolean operations). These linear-time computations only add a small overhead to the global simulation process.

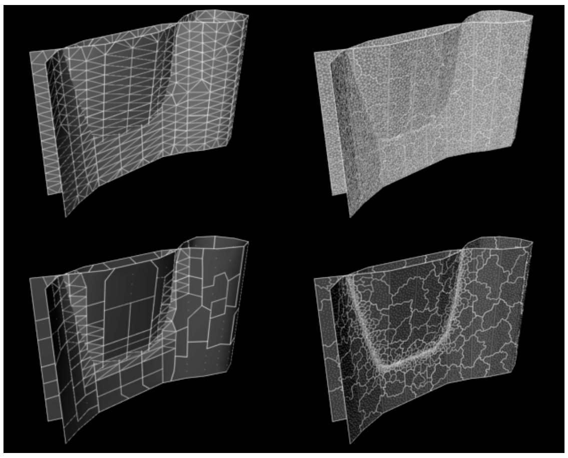


***Figure**: During the detection process, surface regions considered for the detection are similar in size and number whatever the discretization.*

The proposed scheme also behaves very well for highly discretized surfaces, as extra discretization usually yields flatter surfaces relatively to the size of the mesh elements. Hence, unless being near a collision area, the area of the surface regions considered during the detection process always have similar sizes whatever their discretization.

The major benefit of the curvature-based hierarchical collision is that full performance of hierarchical bounding-volume collision detection is preserved for self-collision detection, as the computation is not crippled by detecting all the "colliding" adjacent mesh elements of the surface.

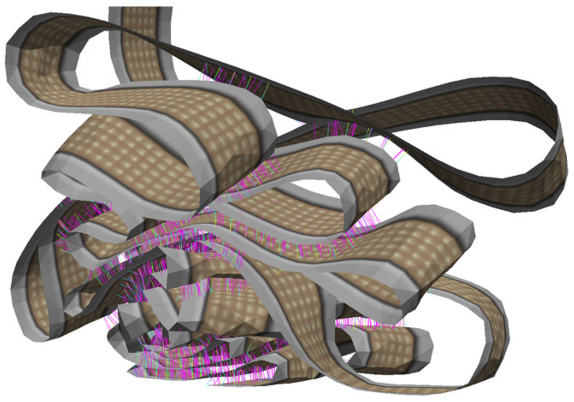


***Figure**: Cloth simulation is heavily relying on self-collision detection.*

4. Collision Response

Once the colliding polygons of the mesh have been extracted, they need to be expressed as geometric information that carries some meaning on how the surfaces are colliding. Usually, collisions may be sorted out as **intersections**, where two surface elements interpenetrate each other, and **proximities**, where two surface elements are separated by a distance below a given threshold, usually representing the "thickness" of the surface. In the case of collisions between polygonal meshes, we can identify the following cases:

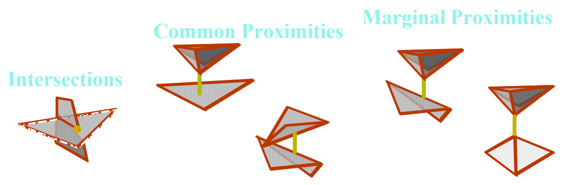


***Figure**: Collision configurations in a polygonal mesh.*

Collision response intends to enforce the fact that real surfaces cannot cross each other. It may either handle intersections usually by backtracking the motion leading to the surface crossing and integrating the collision effect, and proximities by maintaining a minimum separation distance between the surfaces. In either case, the collision effect is usually applied on the mesh vertices of the colliding elements, which carries the geometrical information of the mesh.

4.1. Collision Response Schemes

Collision response has to reproduce reaction and friction effects through adequate action in the ongoing mechanical simulation. Its integration in the mechanical simulation system goes through alteration of the mechanical quantities from the value they would have without the collision effects. There are two main ways for handling collision response:

* **Mechanical response**, where the collision reaction is simulated by forces or by force adjustments which reproduce the contact effect.

* **Geometrical response**, where the collision reaction is simulated by direct corrections on the positions and velocities of the objects.

The mechanical approach is the most formal way of dealing with the problem. The forces or energetic contributions generated from the response can directly be integrated into the mechanical model and simulated. As all the effects are taken into account in the same computation step, the resulting simulation produces an animation where collision response and other mechanical forces add their effects in a compatible way. Reaction is typically modeled by designing a collision penalty force which will repulse the colliding objects from each other and prevent them from intersecting. The repulsion force function is usually designed as a continuous function of the collision distance, and as a piecewise function using simple linear or polynomial intervals. Designing the optimal shape is difficult, because of all these compromises which depend on the actual mechanical context of the simulation. The biggest issue is to model in a robust way geometrical contact (very small collision distance), in which collision response forces only act in a very small range when considered at the macroscopic scale. This implies the use of very strong and rapidly evolving reaction forces, which are difficult to simulate numerically, since a suitable numerical process should discretize the collision contact duration into timesteps that are numerous enough for an accurate reproduction of the collision effects and which cause problem with the usual simulation timesteps which are usually too large.

The geometrical approach aims to reproduce directly the effects of collision response on the geometrical state of the objects without making use of mechanical forces, and thus in a process separated from the mechanical simulation. The advantages are obvious: Geometrical constraints are directly enforced by a geometrical algorithm, and the simulation process is relieved from high intensity and highly discontinuous forces or other mechanical parameters, making it faster and more efficient. This drawback however results from this separation: As collision response changes the geometrical state of the objects separately from the mechanical process, nothing ensures the compatibility of this deformation to a correct variation of the mechanical state that would normally result from it. Furthermore, there is no compatible “additivity” of geometrical variations as there is for forces and energy contributions. The resulting collision effects may be incompatible with mechanics, but also between several interacting collisions. All these issues have to be addressed for providing a collision response model that provides acceptable and steady responses between all the frames of an animation.

Collision effects are decomposed into **reaction effects** (normal components), which are the obvious forces preventing the object penetrating into each other, and **friction effects** (tangential components), which model additional forces that oppose the sliding of objects. The most common friction model it the solid Coulombian friction, where friction forces opposing the motion do not exceed reaction forces times a friction coefficient.

Bibliography

R. BRIDSON, R. FEDKIW, J. ANDERSON, Robust Treatment of Collisions, Contact and Friction for Cloth Animation, ACM Transaction on Graphics, Proceedings of ACM SIGGRAPH, 21(3), pp 594-603, 2002.

J.D. COHEN, M.C. LIN, D. MANOCHA, M.K. PONAMGI, I-COLLIDE: An Interactive and Exact Collision Detection System for Large-Scale Environments, Symp. of Interactive 3D Graphics proc., pp 189-196, 1995.

K. FUJIMURA, H. TORIYA, K. YAMAGUSHI, T.L. KUNII, Octree Algorithms for Solid Modeling, Computer Graphics, Theory and Applications, Proceedings of InterGraphics'83, Springer-Verlag, pp 96-110, 1983.

F. GANOVELLI, J. DINGLIANA, C. O'SULLIVAN, Buckettree: Improving Collision Detection Between Deformable Objects, Proceedings of Spring Conference on Computer Graphics, 2000.

S. GOTTSCHALK, M.C. LIN, D. MANOCHA, OBB-Tree: A Hierarchical Structure for Rapid Interference Detection, Computer Graphics, Proceedings of ACM SIGGRAPH, Addison-Wesley, pp 171-180, 1996.

M. HELD, J.T. KLOSOWSKI, J.S.B. MITCHELL, Evaluation of Collision Detection Methods for Virtual Reality Fly-Throughs, Proceedings of the 7th Canadian Conference on Computational Geometry, 1995.

P.M. HUBBARD, Approximating Polyhedra with Spheres for Time-Critical Collision Detection, ACM Transactions on Graphics, 15(3) pp 179-210, 1996.

P. JIMENEZ, F. THOMAS, C. TORRAS, 3D Collision Detection: A Survey, Computer and Graphics, 25(2), pp 269-285, 2001.

J.T. KLOSOWSKI, M. HELD, J.S.B. MITCHELL, Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs, IEEE transactions on Visualization and Computer Graphics, 4(1), 1997.

T. LARSSON, T. AKENINE-MOLLER, Collision Detection for Continuously Deforming Bodies, Proceedings of Eurographics, pp 325-333, 2001.

M.C. LIN, J.F. CANNY, Efficient Collision Detection for Animation, Proceedings of the Eurographics Workshop on Animation and Simulation, 1992.

M.C. LIN, S. GOTTSCHALK, Collision Detection Between Geometric Models: A Survey, Proceedings of the IMA Conference on Mathematics of Surfaces, 1998.

J. LOMBARDO, M.P. CANI, F. NEYRET, Real-Time Collision Detection for Virtual Surgery, Proceedings of Computer Animation, IEEE Press, pp 82-91, 1999.

J. MEZGER, S. KIMMERLE, O. ETZMUSS, Hierarchical Techniques in Collision Detection for Cloth Animation, Journal of WSCG, 11(2), pp 322-329, 2003.

I.J. PALMER, R.L. GRIMSDALE, Collision Detection for Animation using Sphere-Trees, Computer Graphics Forum, 14, pp 105-116, 1995.

X. PROVOT, Collision and Self-Collision Handling in Cloth Models Dedicated to Design Garments, Proceedings of Graphics Interface, pp 177-189, 1997.

A. SMITH, Y. KITAMURA, H. TAKEMURA, F. KISHINO, A Simple and Efficient Method for Accurate Collision Detection among Deformable Polyhedra, Proceedings of IEEE Virtual Reality Annual International Symposium, pp 136-145, 1995.

G. VANDENBERGEN, Efficient Collision Detection of Complex Deformable Models using AABB Trees, Journal of Graphics Tools, 2(4), pp 1-14, 1997.

P. VOLINO, N. MAGNENAT-THALMANN, Efficient Self-Collision Detection on Smoothly Discretised Surface Animation Using Geometrical Shape Regularity, Computer Graphics Forum (Eurographics'94 proceedings), Blackwell Publishers, 13(3), pp 155-166, 1994.

R.C. WEBB, M.A. GIGANTE, Using Dynamic Bounding Volume Hierarchies to improve Efficiency of Rigid Body Simulations, Communicating with Virtual Worlds, Proceedings of CGI'92, pp 825-841, 1992.

G. ZACHMANN, Rapid Collision Detection by Dynamically Aligned DOP-Trees, Proceedings of IEEE Virtual Reality Annual International Symposium, pp 90-97, 1998.

Collision Detection for Cloth Simulation

Stefan Kimmerle Johannes Mezger

WSI/GRIS, University of Tübingen, Germany

Abstract

In the simulation of cloth, collision detection is crucial for the quality of the results and for the performance as the collision detection process can be highly complex. Contrary to volumetric bodies, the accuracy requirements for the collision treatment of textiles are particulary strict as any overlapping is visible. Additionally cloth tends to self-intersections which also have to be detected efficiently.

In this part of the tutorial, we will first discuss the specificity of collision detection for clothes and will thereafter review some recently used techniques for the real-time collision detection of clothes. In the last section we will detail a fast collision detection approach based on bounding volume hierarchies as it is specially suited both for animated as for static scenes and is also able to detect self-collisions.

1. Introduction

In the physically-based simulation of cloth, collision detection is an essential component. Compared to collision detection for rigid objects, collision detection for cloth shows various aspects that complicate the detection process. Collision detection for cloth objects not only is the detection of intersections between the cloth and an avatar but it also has to take into account the low bending resistance of textiles and therewith the possibility of self-intersections. Additionally, as detailed in the next section, many collision detection approaches for rigid objects are accelerated by spatial data structures like bounding-box hierarchies, distance fields or other kinds of spatial partitioning. These object representations are commonly built in pre-processing steps and therefore perform very well for rigid objects. However for deformable objects like cloth or animated avatars it is necessary to frequently update these data-structures. Another important demand on collision detection for cloth is the supply of sufficient information usable for collision response. Often it is not enough to only detect the interference of two objects, but precise information such as penetration depth or direction is needed.

2. Techniques for collision detection on deformable models

In this section we will first give an overview over previous collision detection methods both for rigid an for deformable objects. Later we will focus on fast collision detection approaches for cloth.

Numerous collision detection methods for various purposes have been developed [18, 28]. Most of the work however focusses on convex polyhedra [17, 21], respectively methods for decomposing non-convex objects into convex parts [6]. However, these approaches are not applicable for highly non-convex objects such as cloth. Currently, collision detection for non-convex objects is mostly accelerated by bounding volume hierarchies. As detailed in section 3.1 the bounding volumes applied range from spheres, oriented bounding boxes (OBBs), axis aligned bounding boxes (AABBs) to shell trees.

Most of these approaches, however, are not designed for deformable models as they rely on pre-computed data. Various researchers have tried to circumvent these problems and to adapt the methods to deformable objects by efficiently updating or rebuilding AABBs [29, 8]. It appears that AABBs perform better than more complex bounding volumes like OBBs when dealing with highly deformable objects since they can be faster updated. It has also been pointed out that higher order trees like 4-ary trees or 8-ary trees are more efficient than binary trees for deformable objects with many intersecting or close areas like in the case of cloth or human tissue, because fewer nodes need to be updated in each time step and the recursion depth during overlap tests is lower [16, 20].

Besides bounding volume hierarchies recent collision detection methods for deformable objects have relied on distance fields [5, 7], graphics hardware [2, 19, 30, 12], or spatial subdivision methods, dividing the scene into voxels [35, 27]. These approaches will be detailed in the following subsections.

Methods based on curvature and orientation to accelerate self-collision detection have also been presented for cloth simulation [32, 25]. Self-intersection needs special attention, since thin objects are subject to tunnelling effects due to time sampling. To avoid the tunnelling a continuous motion from the start to the end of each time step has to be considered as a whole rather than considering initial and final states separately. Continuous collision detection has been studied for both deformable [4] and rigid bodies [26].

In the following we detail distance fields and image-space techniques, that where employed for fast collision detection of cloth objects.

2.1. Distance Fields

Distance fields as shown in Fig. 1 specify for each point the minimum distance to a closed surface. To distinguish between inside and outside, a signed distance can be used. For the collision detection step, the evaluation of distances and normals is very fast and therefore even for complex objects allows interactive frame rates. Nevertheless the computation of the distance field itself is very costly and is usually done in a pre-processing step. Because for animated scenes the distance fields need to be updated in each step, distance fields where often only used for cloth simulations combined with a static environment. In [5] multiple rigid bodies with pre-computed distance fields are combined to build animations. In [30] an image-based approach is used to calculate distance values by constructing depth and normal maps of the collision object. Fuhrmann et al. [7] also show a very efficient implementation of collision detection for cloth simulations with distance fields.

2.2. Image-Space Techniques

In recent years image-space techniques have been proposed for collision detection [23, 3]. These approaches commonly use projections of objects to detect collisions between them. Vassilev et al. [30] present an image-space based collision detection for cloth simulations. In this approach, an avatar is rendered from a back and a front view to generate an approximate representation of its volume, which is used to detect intersecting cloth particles. Fig. 2 shows the algorithm of Heidelberger et al. [11], that is similar to other commonly used approaches. In a first step the intersection of AABBs is calculated. In step (b) a layered depth image is computed that is used in (c) to compute the intersecting volumes of the two objects. Note that image-space techniques work with discretized representations of objects and therefore cannot provide exact collision information like the penetration depth.

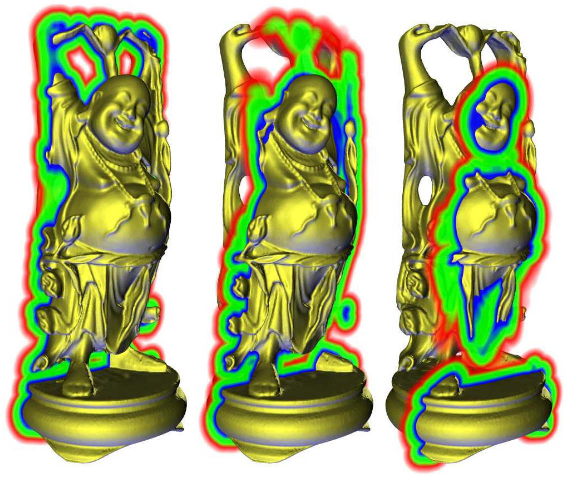
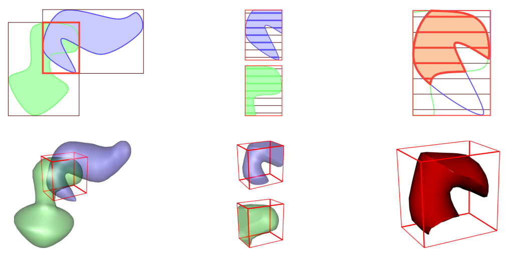


Figure 1: *Happy Buddha and three color-mapped distance field slices. Since the distance field is only valid within a band near the surface, the mapping is faded out at larger distance. Blue maps to close distances, whereas red indicates medium distances.* Images by courtesy of A. Fuhrmann, IGD Darmstadt [7].

Nevertheless they can be used to achieve interactive collision detection for deformable objects and even for self-collision detection as detailed in [12].



(a) Stage 1. (b) Stage 2. (c) Stage 3.

Figure 2: *Image-space collision detection in 2-D and 3-D. (a) AABB intersection. (b) LDI generation within the Volume of Interest. (c) Computation of the intersection volume.* Images by courtesy of B. Heidelberger and M. Teschner, ETH Zürich [12].

3. Bounding Volumes Hierarchies

The basic geometric elements for a collision detection algorithm are triangles or more general polygons of our virtual objects in the considered scene. Every surface involved in the simulation process, being part of the cloth itself or being part of the scenery, can also be assumed to be triangulated.

For collision detection we basically have to make distance computations for each possible pair of triangles. Obviously this yields an unrealistic task making up a so called *complexity problem*.

A lot of possible solutions how to break down this complexity problem have been proposed in recent years. However they all have the common idea, not to test all the triangles. To find the triangles to be tested, complex objects are embedded into *bounding volumes* as described in the next subsection, and, to avoid checking triangles far away from each other, grids or hierarchical structures are used as described in the subsection thereafter.

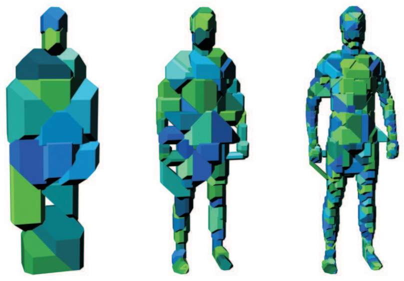


Figure 3: *Three levels of an 18-DOP-hierarchy.*

3.1. Bounding Volumes

Reducing complexity by employing fast and few tests for complex objects is the basic idea of bounding volumes. Moore et al. [22] already introduced the idea of bounding volumes (BV) in 1988. These bounding volumes are geometric objects enclosing complex objects or primitives. The collision detection process is then first performed on the bounding volumes. Only if an intersection between the bounding volumes is detected, the objects inside the bounding volumes are tested for intersections in the second step.

In the past, a wealth of BV types has been proposed, such as spheres [13, 24], oriented bounding volumes (OBB) [9], discrete oriented polytopes (DOP) [14, 33], boxtrees [34, 1], axis aligned bounding boxes (AABB) [29, 16], spherical shells [15], and convex hulls [6].

When choosing adequate BVs the corresponding advantages and disadvantages for the specific case have to be considered. It is important, how fast the collision test between two BVs is, how well the BV fits the object, if a lot of storage is needed to describe an individual bounding volume and if the construction and update of the BV is efficient. For the simulation of cloth, the highly non-convex shape of the object and the continuous deformations have to be considered. Because these arbitrary object deformations cannot be expressed by rigid rotations and translations, oriented bounding volumes like OBBs [9], QuOSPOs [10], or Dynamically Aligned DOPs [33] are not suitable for cloth simulation.

In the following, we follow the ideas of Mezger et al. [20] and therefore choose common k-DOPs instead of AABBs or spheres as they show better convergence while still being easy to check for intersections. By convergence we mean the hierarchical series of bounding volume approximation of the surface (or object) to be checked.

3.1.1. *k*-DOPs

A *k*-DOP[14] (discrete oriented polytope) is a convex polyhedron defined by *k* halfspaces denoted as

$$H_i = \{x \in \mathbb{R}^m \mid n_i^T x \le b, n_i \in N, b_i \in \mathbb{R}\}.$$

The normals n_i of the corresponding hyperplanes of all k-DOPs are discrete and form the small set $N = \{n_1, \dots, n_k\} \subseteq \mathbb{R}^m$. For arithmetic reasons the entries of the normal-vectors are usually chosen from the set $\{-1, 0, 1\}$. In order to turn the intersection tests for the polyhedrons into simple interval tests, the hyperplanes have to form $k/2$ parallel pairs. E.g., an axis aligned bounding box (AABB, 6-DOP) in $\mathbb{R}^3$ is given by $N_6 = \{(\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1)\}$, an octahedron (8-DOP) is generated by setting all normal components to ± 1. We usually use 14-DOPs ($N_{14} = N_6 \dot{\cup} N_8$) or 18-DOPs (AABB with clipped edges).

The easiest way to build the k-DOP bounding volume for a set of points is inserting them into a primarily empty k-DOP by updating its $k/2$ intervals accordingly. The overlap test between two k-DOPs is implemented by interval tests similar to the common AABB, indicating disjointness as soon as one pair of intervals is disjoint. Thus, the maximal number of interval tests is $k/2$ (in the overlapping case).

3.2. Hierarchies and Spatial Subdivision

3.2.1. *k*-DOP-Hierarchy Generation

Let BV_k be the tightest k-DOP enclosing a set of vertices and $\widetilde{\bigcup}$ the operation forming the tightest k-DOP enclosing out of a set of k-DOPs. Then, like AABBs also k-DOP bounding volumes satisfy the equation

$$BV_k(V) = \widetilde{\bigcup}_{p \in P(V)} BV_k(p) \qquad (1)$$

for a set of vertices V and an arbitrary partition $P(V)$. Hence, the optimal bounding volume for a node in the hierarchy can be easily computed by merging its child bounding volumes. Vice versa the hierarchy can be efficiently built using a top-down splitting method. Figure (3) shows three hierarchy levels for the 18-DOP-hierarchy of an avatar.

3.2.2. Node split and node merge

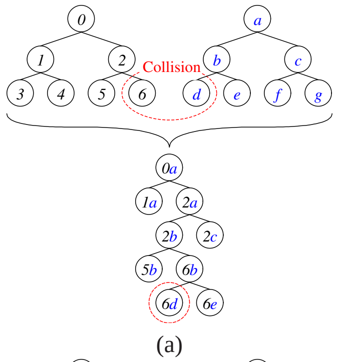
As for octrees, to build the hierarchy of bounding volumes, each parent node need to be split into smaller parts if the hierarchy is built top-down. Otherwise, if the hierarchy is built bottom-up, specific child nodes need to be merged to a parent node.

Different methods have been presented to perform the splitting process in the first case. For AABBs and rigid objects, Zachmann [34] shows that any splitting heuristic should minimize the volume of the children nodes by analyzing the Minkowski sums of the corresponding BVs. Mezger et al. [20] simply split the k-DOPs according to the longest side. The longest side of a k-DOP is determined by the face pair with the maximum distance. The k-DOP is split parallel to this face pair through its center. As generally some polygons are cut by the splitting plane, they are assigned to that child node which would contain the smallest number of polygons. In the lower hierarchy levels, if all polygons happen to be cut, each of them is assigned to its own node. Finally, as the corresponding vertices for the node are known, the k-DOPs can be optimally fitted to the underlying faces. Although this method is simple, it turns out to be efficient on the one hand and to produce well balanced trees on the other hand. The complete hierarchy setup for objects holding several thousands of polygons can be performed within merely a second, allowing to add objects dynamically to the scene.

Volino and Magnenat-Thalmann [32, 31] as well as Provot [25] use a fairly different approach to built the hierarchy for cloth objects. In their methods, the hierarchy is strictly oriented on the mesh topology of the object, assuming that topology does not change during the simulation. Volino uses a region-merge algorithm to build the hierarchy bottom-up, while Provot uses a top-down algorithm that recursively divides the object in zones imbricating each other. These approaches have the advantage, that they avoid grouping faces close together in the hierarchy that are very close in the initial state, although they are not close at all based on the connectivity. This connectivity-based approach also yields advantages in speeding-up self-collision detection.

3.2.3. Trees

Another crucial point is the arity of the bounding volume hierarchy. For rigid objects, binary trees are commonly chosen. In contrast, 4-ary trees or 8-ary trees have shown better performance for deformable objects [16, 20]. This is mainly due to the fact that fewer nodes need to be updated and the total update costs are lower. Additionally, the recursion depth during overlap tests is lower and therefore the memory requirements on the stack are lower. Fig. 4 shows the reduction of recursion depth for detecting two overlapping leaves by equivalent 4-ary trees instead of binary trees. However, if only the root nodes overlap in the example, the 4-ary trees require four overlap tests, which is two times more than using binary trees. Since overlap tests for k-DOPs only need $k/2$ interval tests in the worst (overlapping) case, a slight increase of overlap tests is acceptable.



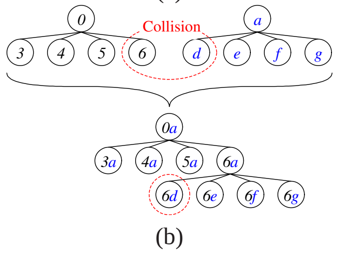


Figure 4: *Recursion using binary trees (a) and 4-ary trees (b).*

3.2.4. Hierarchy Update

In contrast to hierarchies for rigid objects, hierarchies for cloth objects need to be updated in each time step. Principally, there are two possibilities: updating or rebuilding. Refitting is much faster than rebuilding, but for large deformations, the BVs usually are less tight and have larger overlap volumes. Nevertheless, van den Bergen [29] has found that refitting is about ten times faster compared to a complete rebuild of an AABB hierarchy. Further, as long as the topology of the object is conserved there is no significant performance loss in the actual collision query compared to rebuilding.

Different strategies have been proposed not only for building a hierarchy but also for the hierarchy update. Larsson and Akenine-Möller [16] compared bottom-up and top-down strategies. They found that if in a collision detection process many deep nodes are reached the bottom-up strategy performs better, while if only some deep nodes are reached the top-down approach is faster. Therefore they proposed a hybrid method, that updates the top half of the tree bottom-up and only if non-updated nodes are reached these are updated top-down. Using this method they reduce the number of unnecessarily updated nodes with the drawback of higher memory requirement because they have to store the leaf information about vertices or faces also in the internal nodes.

Other approaches have been proposed by Mezger et al. [20] to further accelerate the hierarchy update by omitting or simplifying the update process for several time steps. For this purpose the bounding volumes can generally be inflated by a certain distance. Then the hierarchy update is not needed as long as the enclosed primitives did not move farther than that distance.

3.3. Hierarchy traversal and Self-Collision Heuristics

The above described collision detection technique is now applied to an actual cloth surface. For this we consider each polygon of our cloth-surface as a separate, individual object participating in the simulation. Upon these objects, together with the objects the cloth may collide with, we build the hierarchical structure described above. While traversing the hierarchies top-down to the individual polygons in the leaves, we may process the collisions regardless if the corresponding polygons belong to different objects or to the same (cloth) object.

Additionally to this standard procedure, for self-collision of the cloth object auxiliary information can be used to speed up the computation. The two corresponding ideas are that adjacent polygons can always be excluded from self-collision detection, and that a flat surface cannot have self-intersection at all. To cause self-intersections the object must be strongly bend. An exact method to reject possible self-intersections for a certain region was suggested by Volino and Magnenat-Thalmann [32], where a vector is searched that has positive dot product with all normals of the region. If such a vector exists and the projection of the region onto a plane in direction of the vector does not self-intersect, the region cannot self-intersect either.

Provot [25] also proposed a method, which is very fast and accurate enough for regions having a sufficiently convex border. For every region a cone is maintained representing a superset of the normal directions. The cone can be calculated during the bottom-up hierarchy update by very few arithmetic operations. The apex angle α of the cone represents the curvature of the region, indicating possible intersections if $\alpha \geq \pi$.

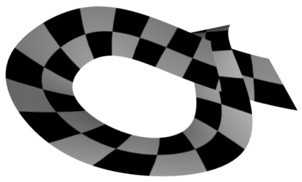


Figure 5: *Self-intersecting mesh with correlated normals but concave shape.*

Still there remains the problem that hierarchy regions can have severe non-convex shape and therefore compromise the robustness of the surface curvature criterion. Figure (5) shows such a surface that self-intersects although the apex angle of its normal cone is rather small. We divide such a mesh into several face groups and build an adjacency matrix for the groups. The curvature heuristic is not applied to non-adjacent groups during the self-collision test.

4. Conclusions

We described the difficulties and demands on collision detection methods for cloth simulation. Moreover, we discussed and compared several collision detection approaches that fulfill these demands. In the last section we detailed one of these approaches based on a hierarchy of *k*-DOPs while addressing the problems specific to cloth like self-collisions and fast hierarchy updates. This method is widely used for the collision detection in the animation of cloth and dynamic avatars.

References

[1] P.K. Agarwal, M.T. de Berg, J.G. Gudmundsson, M. Hammar, and H.J. Haverkort. Box-trees and r-trees with near-optimal query time. *Discrete and Computational Geometry*, 28(3):291–312, 2002.

[2] G. Baciu, W. Wong, and H. Sun. Hardware-Assisted Virtual Collisions. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST, Taipei, Taiwan*, pages 145–151, 1998.

[3] G. Baciu, W. Sai-Keung Wong, and H. Sun. RECODE: an image–based collision detection algorithm. *The Journal of Visualization and Computer Animation*, 10:181–192, 1999.

[4] R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. *ACM Transactions on Graphics*, 21(3):594–603, 2002. Proc. ACM SIGGRAPH 2002.

[5] Robert Bridson, S. Marino, and Ronald Fedkiw. Simulation of clothing with folds and wrinkles. In *Proc. ACM/Eurographics Symposium on Computer Animation*, pages 28–36, 2003.

[6] Stephan A. Ehmann and Ming C. Lin. Accurate and fast proximity queries between polyhedra using convex surface decomposition. In *Computer Graphics Forum*, volume 20, pages 500–510, 2001. ISSN 1067-7055.

[7] Arnulph Fuhrmann, Gerrit Sobotka, and Clemens Gross. Distance fields for rapid collision detection in physically based modeling. In *Proceedings of GraphiCon 2003*, pages 58–65, September 2003.

[8] F. Ganovelli, J. Dingliana, and C. O'Sullivan. Buckettree: Improving collision detection between deformable objects. In *Proc. of Spring Conference on Computer Graphics SCCG '00*, 2000.

[9] Stefan Gottschalk, Ming Lin, and Dinesh Manocha. OBB-Tree: A hierarchical structure for rapid interference detection. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, pages 171–180. ACM SIGGRAPH, Addison Wesley, August 1996.

[10] Taosong He. Fast Collision Detection Using QuOSPO

Trees. *Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 55–62, 1999.

[11] B. Heidelberger, M. Teschner, and M. Gross. Real-time volumetric intersections of deforming objects. In *Proc. of Vision, Modeling, Visualization VMV'03*, pages 461–468, 2003.

[12] B. Heidelberger, M. Teschner, and M. Gross. Detection of collisions and self-collisions using image-space techniques. In *Proc. of WSCG'04*, 2004.

[13] Philip M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics*, 15(3):179–210, July 1996.

[14] James T. Klosowski, Martin Held, Joseph S. B. Mitchell, Henry Sowizral, and Karel Zikan. Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998.

[15] S. Krishnan, M. Gopi, M. Lin, D. Manocha, and A. Pattekar. Rapid and accurate contact determination between spline models using ShellTrees. *Computer Graphics Forum*, 17(3), September 1998.

[16] Thomas Larsson and Tomas Akenine-Möller. Collision detection for continuously deforming bodies. In *Eurographics*, pages 325–333, 2001. short presentation.

[17] M. C. Lin and J. F. Canny. Efficient Collision Detection for Animation. In *Proc. 3rd Eurographics Workshop on Animation and Simulation*, 1992.

[18] M. C. Lin and S. Gottschalk. Collision Detection Between Geometric Models: A Survey. *Proc. of IMA Conference on Mathematics of Surfaces*, 1998.

[19] J.C. Lombardo, M.-P. Cani, and F. Neyret. Real-time Collision Detection for Virtual Surgery. In *Proc. Comp. Anim. '99*, pages 82–91. IEEE CS Press, 1999.

[20] Johannes Mezger, Stefan Kimmerle, and Olaf Etzmuß. Hierarchical Techniques in Collision Detection for Cloth Animation. *Journal of WSCG*, 11(2):322–329, 2003.

[21] B. Mirtich. VClip: Fast and Robust Polyhedral Collision Detection. *ACM Transactions on Graphics*, 17(3):177–208, 1998.

[22] M. Moore and J. Wilhelms. Collision detection and response for computer animation. volume 22 of *Annual Conference Series*, pages 289–298. ACM SIGGRAPH, July 1988.

[23] K. Myszkowski, O. Okunev, and T. Kunii. Fast collision detection between complex solids using rasterizing graphics hardware. *The Visual Computer*, 11(9):497–512, 1995.

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

[25] Xavier Provot. Collision and Self-Collision Handling in Cloth Model Dedicated to Design Garments. In *Graphics Interface '97*, pages 177–189, 1997.

[26] Stephane Redon, Abderrahmane Kheddar, and Sabine Coquillart. Fast continuous collision detection between rigid bodies. *Computer graphics forum*, 21(3), 2002. Proc. Eurographics'02.

[27] M. Teschner, B. Heidelberger, M. Mueller, D. Pomeranets, and M. Gross. Optimized spatial hashing for collision detection of deformable objects. In *Proceedings of Vision, Modeling, Visualization VMV'03*, pages 47–54, 2003.

[28] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, and P. Volino. Collision detection for deformable objects. State of the Art Report, Eurographics 2004.

[29] Gino van den Bergen. Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics Tools: JGT*, 2(4):1–14, 1997.

[30] T. Vassilev, B. Spanlang, and Y. Chrysanthou. Fast Cloth Animation on Walking Avatars. In *Computer Graphics Forum (Proc. of Eurographics)*, 2001.

[31] P. Volino and N. Magnenat-Thalmann. Collision and Self-Collision Detection: Efficient and Robust Solutions for Higly Deformable Surfaces. In *Comp. Animation and Simulation*, pages 55–65. Springer Verlag, 1995.

[32] Pascal Volino and Nadia Magnenat-Thalmann. Efficient Self-Collision Detection on Smoothly Discretized Surface Animations using Geometrical Shape Regularity. *Computer Graphics Forum*, 13(3):155–166, 1994.

[33] Gabriel Zachmann. Rapid collision detection by dynamically aligned DOP-trees. In *Proc. of IEEE Virtual Reality Annual International Symposium; VRAIS '98*, pages 90–97, Atlanta, Georgia, March 1998.

[34] Gabriel Zachmann. Minimal hierarchical collision detection. In *Proc. ACM Symposium on Virtual Reality Software and Technology (VRST)*, pages 121–128, Hong Kong, China, November11–13 2002.

[35] Dongliang Zhang and Matthew M.F. Yuen. Collision Detection for Clothed Human Animation. *Proceedings of the Pacific Graphics*, 2000.

Acquisition and Real-Time Rendering of Textile

J. Meseth, G. Müller, M. Sattler, R. Sarlette and R. Klein

University of Bonn, Institute for Computer Science II

Abstract

Producing fast, high-quality visualizations of cloth is still a major challenge in computer graphics due to the complex meso structure and reflectance behavior of fabrics which define the unique look-and-feel of a material. A wide class of such realistic materials can be described as 2D-texture under varying light- and view direction namely the Bidirectional Texture Function (BTF). Since an easy and general method for modeling BTFs seems out of reach, current research concentrates on image-based methods which rely on sampled BTFs (acquired real-world data) in combination with appropriate synthesis methods. Recent results have shown that this approach greatly improves the visual quality of rendered surfaces and therefore the quality of applications such as virtual prototyping.

This tutorial presents an overview over state-of-the-art techniques for the main tasks involved in producing photo-realistic renderings using measured BTFs and provides in-depth descriptions of selected approaches.

1. Introduction

The visualization of realistic materials and object surfaces is still one of the main challenges in todays computer graphics.

Traditionally the geometry of a surface is modeled explicitly (e.g. with triangles) only up to a certain scale while the micro structure responsible for the reflectance behavior of a material is simulated using relatively simple mathematical models. For special groups of materials such as metals or shiny plastic these models are able to generate a more or less correct approximation of reality, but they are not suited and even designed for as complex materials as fabrics. Furthermore these models are not easy to parameterize in order to achieve a desired effect, because the parameters do not correspond to some intuitive or physical meaning.

Things become even more complicated, if one wants to simulate the so called meso structure of a complex inhomogeneous material. These structures are in-between the macro-scale geometry modeled by triangles and the micro-scale geometry modeled mathematically. Complex meso structures are often too fine to be modeled by triangles, but they are also too big to be captured in a simple mathematical reflectance model, since they are visible to the viewer. And unfortunately the meso-structure plays a very important role in defining the unique look of a material, hence the weaknesses of traditional modeling in visualizing complex meso structure are a major drawback for photo-realistic image synthesis.

Promoted by the limitations of traditional modeling, data-driven approaches have gained a growing amount of attention in the last few years not only in the computer graphics community but also in other areas such as synthesis of human speech. In computer graphics these methods fall into the field of image-based rendering (e.g. texture mapping is a well known and widely used image-based technique for creating the impression of complex meso structure). But unfortunately the correct impression can only be recreated for flat surfaces with simple albedo variation, while texture mapping rough surfaces is only valid for the unique viewing and lighting condition under which the texture image was taken. Obviously the general appearance of a complex surface can not be captured by a single 2D-image.

As illustrated in figure 2 the full variability of the interaction of light with an arbitrarily complex time-variant 3-dimensional surface can be descibed by a 12-dimensional function.

Since definition or measurement of such a function is not practical, some assumptions about this function have to be made:

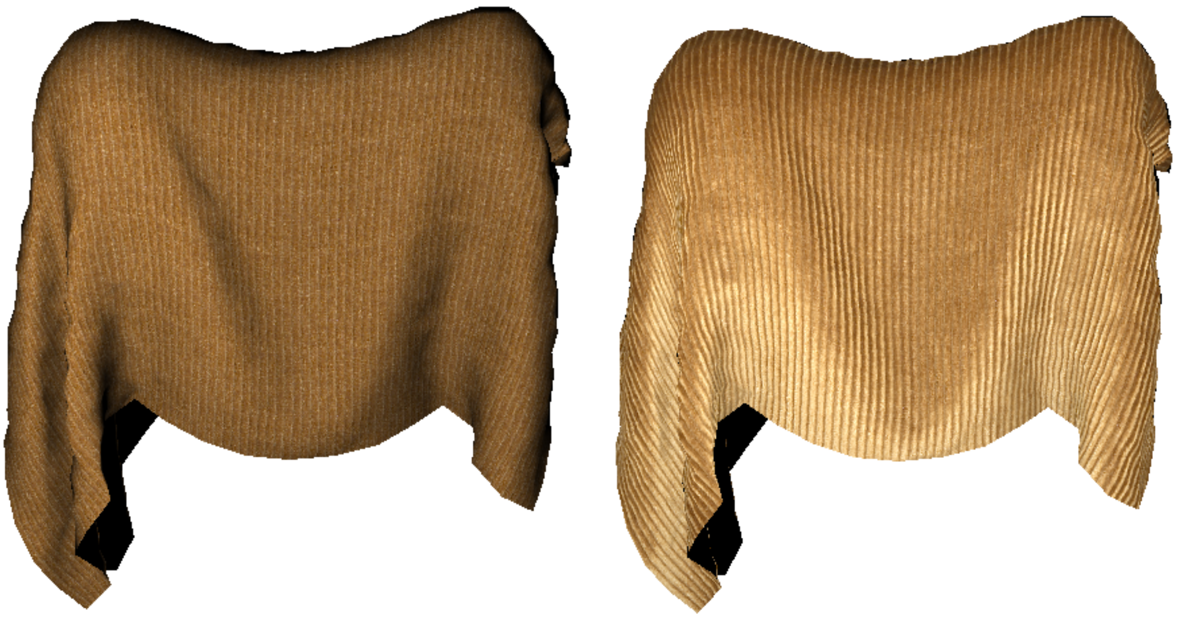


Figure 1: *Comparison of diffuse textures (left) and BTF rendering (right). One can clearly see the increased realism resulting from more realistic reflectance behaviour and increased depth impression in the right image.*

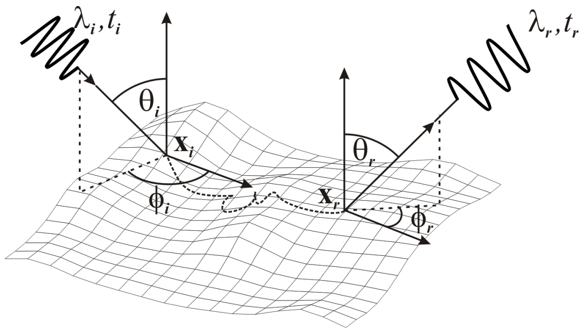


Figure 2: *The parameters of general light-material interaction*

- no phosphorescence (light leaves material immediately $t_i = t_r$)
- temporal invariance (consider only a fixed point t_0 in time)
- no fluorescence (wavelength is not changed $\lambda_i = \lambda_r$)
- discretization of wavelength into the three color bands red, green and blue (consider only $\lambda_{r,g,b}$)

We now have arrived at an 8-dimensional function, which is called the bidirectional surface scattering distribution function (BSSRDF) [NRH*77] and describes the light transport between every point on the surface for any incoming and outgoing direction. Since spatially dependent subsurface scattering effects are rather low in most materials used for clothes, two additional dimensions can be dropped without introducing large errors into realistic cloth visualization systems. This six-dimensional function is called a Bidirectional Texture Function (BTF) and captures view- and light-dependence of an inhomogeneous surface. Sampling this function consists of taking images under varying light and viewing conditions. Renderings from this function can be generated e.g. by linear interpolation between the samples. As illustrated in figure 1 the visual impression of complex materials is improved significantly compared to simple texture mapping and diffuse relighting.

In the following sections, efficient methods for acquisition and rendering of complex materials like cloth are presented. While many methods are overviewed just very briefly, selected methods are presented in greater detail, optimally providing sufficient information for direct implementation.

2. BTF Acquisition

A first step towards high-quality acquisition of real-world materials was made by Dana et al. [DvGNK97] some years ago already. Their work resulted in a data set of 61 samples of real-world surfaces publicly available in the CUReT [CUR] database. The database was created to investigate the visual appearance of real-world surfaces consisting of spatially varying, isotropic materials, but does not satisfy all of the requirements for high quality cloth rendering: first, some acquired images contain graphics errors caused by framegrabber artifacts or reflections of the robot sample holder plate visible in the raw data. Second they lack angular resolution, i.e. too few lighting and viewing angles were covered during the measurement step in order to faithfully reproduce the complex reflectance behavior of cloth. Third, unfortunately the data is not spatially registered which makes public use very difficult. Nevertheless their measurement setup strongly influenced further acquisition systems.

McAllister [McA02] implemented an improved setup capable of acquiring anisotropic BTFs by taking digital images for selected light- and view directions. Independently from the approach of McAllister, a very similar one was taken in Hauth et al. [HEE*02], which will be described in further detail in the next section.

In the context of rendering Surface Light Fields (i.e. BTFs for fixed view directions), Malzbender et al. [MGW01] constructed a measurement device using a number of light sources (which can be switched on and off individually) positioned on the hemisphere around the acquired object. The device allows very fast acquisition since neither the camera nor the light source has to be repositioned. In addition, such a measurement device can be precalibrated very accurately, resulting in very accurate measurements.

Finally, BTF databases can alternatively be created synthetically. Such approaches, which were followed by Daubert et al. [DLHS01], Daubert and Seidel [DS02], Gröller et al. [GRS96] and Liu et al. [LYS01], lead to BTFs with unknown resemblance to existing materials and will therefore not be detailed in this tutorial.

2.1. Measurement Setup

The setup presented in this section was first described in [HEE*02]. It is mainly composed of a lamp with fixed position, a camera mounted on a wagon which resides on a rail system forming a half-circle with radius 170 cm around the probe holder which is mounted on a robot, and a personal computer (see figure 3). For the realistic reproduction of cloth appearance, the lamp has to approximate directional sunlight. To achieve this, a Hydrargyrum Medium Arc Length Iodide (HMI)-light source (bron imaging HMI F575), which is widely used in the professional film- and photo-industry, is used in combination with a Fresnel-lens. As capturing device a programmable high resolution CCD-camera (Kodak DCS Pro 14N) with a maximum resolution of 4000×3500 pixel is utilized. The positioning of the sample is done by a robot arm (Intelitek SCORBOT-ER 4u). All these parts are controlled by a personal computer. The probe holder can hold planar material samples with a maximum extent of 10×10 cm which restricts the set of acquirable materials but turns out sufficient for most types like fabrics, wallpapers, tiles and even car interior materials.

To reduce the influence of scattered light, the lab is darkened and the hardware used for measurements is covered with dark casings. This computer controlled setup allows the automatic reproduction of every constellation of the view/light-direction with reference to the sample surface. See figure 4 for a sketch of the measurement setup.

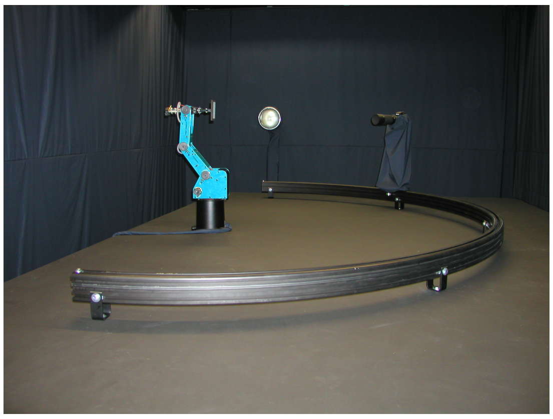


Figure 3: *Picture of the laboratory. The black casings of the robot and light source are removed.*

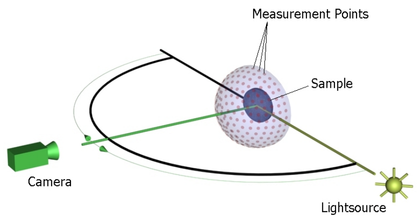


Figure 4: *Illustration of the measurement setup. A fixed light source and different camera positions are shown.*

2.2. Measurement Procedure

To achieve a sufficient measurement quality for rendering, the equipment has to be calibrated.

- To compensate the positioning error due to the robot and the railsystem one has to track the sample holder mounted on the robot arm using the camera. Experiments determined that these errors are very small in the described setup. Therefore, marker points, which are placed on the sample holder, are detected only during the post-processing phase, allowing a software jitter correction.
- A geometric calibration [Rob96] has to be applied to the camera to reduce geometric distortion, which is caused by the optical system of the camera.
- For each sample to be measured, the aperture of the camera gets adjusted in such a way that the number of saturated or dark pixels in the pictures is minimized given a fixed aperture during the measurement process.

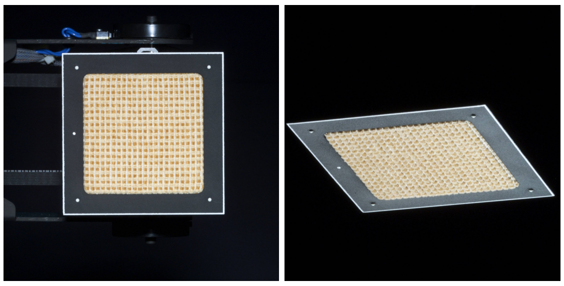


Figure 5: *Sample holder with the PROPOSTE sample. The left image shows the frontal view ($\theta = 0^\circ$, $\phi = 0^\circ$); the right image shows ($\theta = 60^\circ$, $\phi = 342^\circ$). White point and border markers are clearly visible.*

- To achieve the best possible color reproduction, the combination of the camera and the light source has to be color calibrated [DvGNK99]. For the measurement of the camera color profile a special CCD-Camera standard card (Gretag Macbeth - Color Checker DC) is used.

After calibration, a set of view/light directions has to be chosen for measurement. In the remaining text, a view or light direction will be identified by the tupel (θ, ϕ) of polar and azimuth angles with indices v for view and l for light directions. To achieve a consistent sampling density over the hemisphere, $\Delta\theta = 15^\circ$ for each of the viewing and illumination directions and different $\Delta\phi$ as shown in Table 1 are chosen, resulting in different numbers of images per θ segment. The resulting sampling consists of 81 viewing and 81 illumination directions or 6561 images for all combinations which turned out sufficient for most measured samples but optimally should be adjusted to the reflectance properties of the sample materials (e.g. using an approach similar to the one of Lensch et al. [LLSS03]). For practical reasons constallations with $(\theta_l, \phi_l) = (\theta_v, \phi_v)$ cannot be measured. To approximate this missing data, in those cases the varied light direction $(\theta_l - 10^\circ, \phi_l)$ is employed.

During the measurement for every position of the data-set an image is taken and stored for post-processing. Figure 5 shows examples of such images. Finally, another image of the CCD-Camera standard card has to be taken to detect deviations occurred during the measurement process. The measurement time for a single material probe is about 14 hours, whereas the most time is used for the data transfer from the camera to the host computer.

Storing the 6561 images requires about 80 GB, 12 MB per image, employing the Kodak DCR 12-bit RGB format and lossless compression. In the following sections, we will show how this amount can easily be reduced to handlable sizes.

θ [°]	$\Delta\phi$ [°]	No. of images
0	-	1
15	60	6
30	30	12
45	20	18
60	18	20
75	15	24

Table 1: *Sampling density of viewing and lighting angles of the BTF data sets measured with the described setup.*

2.3. Postprocessing

After the measurement the raw image data is converted into a set of rectified, registered images capturing the appearance of the material for varying light and view directions. Therefore the perspectively distorted images are rectified, registered, cut and sorted (see figure 6 for a schematic overview of the postprocessing process).

Registration is done by projecting all sample images onto the plane which is defined by the frontal view $(\theta = 0, \phi = 0)$. To be able to conduct an automatic registration borderline markers were attached to the sample holder plate, see figure 5. After converting a copy of the raw data to black-and-white (8-bit TIFF), standard image processing tools are used to detect the markers. In following steps the perspective projection matrix (which maps these markers to the position of the markers in the frontal view) is computed and utilized to fill the registered image with appropriate colors (which are currently restricted to common 8-bit RGB).

To convert the 12-bit RGB ERIMM [Ame02] images stored as Kodak DCR images to standard 8 bit RGB file formats, the standard color profiles provided with the Kodak SDK (look and output profile) and camera (tone curve profile) are applied to the image. The most appropriate 8 bit color range is extracted after applying an exposure gain to the converted data. As a fixed focal length is used during one measurement, the maximum effective resolution of the sample holder in the image is 1600×1600 pixels. After all transformations are carried out, all images are rescaled to an equal size of 1536×1536 pixels. Such images are called *normtextures*(N) in the remaining text. Examples of these rectified images are shown in figure 7.

After this postprocessing step, the data amount of 80 GB captured by the camera CCD chip is reduced to roughly 46 GB of uncompressed data. This amount can further be reduced by assuming tileablility of the measured material, manually chosing a region of interest (approximately 550×550 pixels for textured stored in the online database, up to 800×800 in principle), cuting it out and resizing it. To create the final normtextures (256×256 pixels in size for probes in the database, up to about 800×800 in princi-

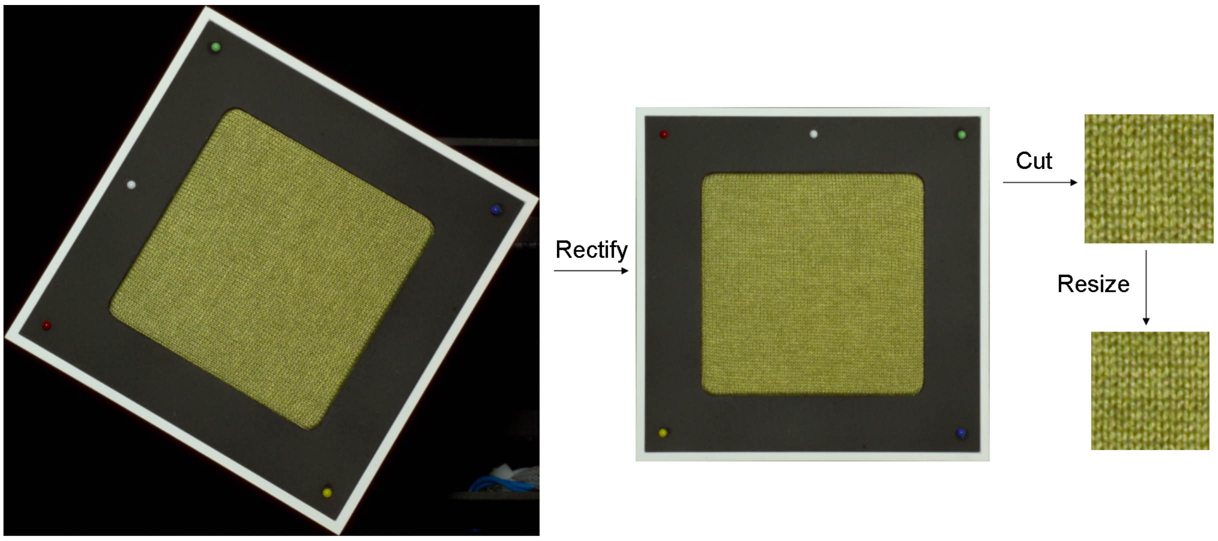


Figure 6: *Postprocessing Pipeline. The left image shows a section from the taken image. The image is rectified, cut and resized for efficient handling.*

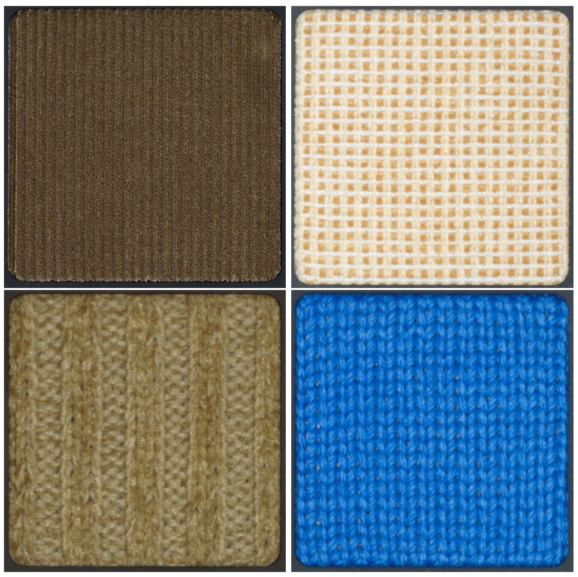


Figure 7: *Measured BTF samples. Top row: corduroy (left), Proposte (right). Bottom row: two kinds of knitted wool.*

ple) linear edgeblending is applied, which reduces the usual tiling artifacts.

Some of the measurements done with the described setup are publicly available from the Bonn BTF Database Bonn [BTF]. Data sets consist of only slightly compressed JPEG images and require between 320 and 360 MB of storage.

3. BTF Compression

Real-time rendering of the raw data from the BTF database consisting of at least 1.2 GB uncompressed data per sample is impractical considering a typical scene with several materials. Thus some sort of data-compression has to be applied.

Existing BTF compression techniques interpret the BTF as in figure 8: as a collection of discrete textures

$$\mathbf{BTF}_{Tex} := \left\{ T_{(\mathbf{v},\mathbf{l})} \right\}_{(\mathbf{v},\mathbf{l})\in\mathcal{M}},$$

where $\mathcal{M}$ denotes the set of discrete measured view- and light-directions, or as a set of tabulated apparent BRDFs.

$$\mathbf{BTF}_{Brdf} := \left\{ B_{(x,y)} \right\}_{(x,y)\in I\subset N^2}.$$

Note, that apparent BRDFs do not fulfill physically demanded properties like reciprocity and furthermore contain a factor $(\mathbf{n}\cdot\mathbf{l})$ between incident direction and surface normal. This is nicely illustrated in figure 8.

A compression technique regarding the BTF as a set of view-dependent textures was proposed by Sattler et al. [SSK03]: they apply principal component analysis (PCA) to the view-dependent textures to determine so-called Eigen-Textures. While this representation can be rendered efficiently using graphics hardware, especially since graphics hardware capabilities like filtering and mipmapping can be used in a straightforward way, the compressed data still requires about 120 MB per material, rendering the method unusefull for most scenes.

All other compression techniques interpret the BTF as

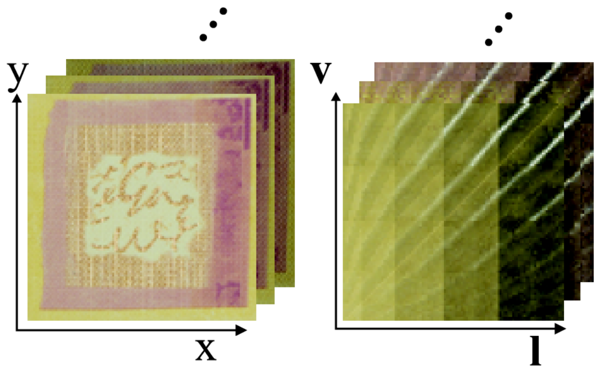


Figure 8: *Two arrangements of the BTF data: As set of images (left) and as set of apparent BRDFs (right).*

spatially varying apparent BRDFs. The techniques can be grouped into three categories: those that fit analytic models and those using linear or nonlinear basis decomposition.

3.1. Fitting Analytic Models

Methods fitting analytic per-pixel apparent BRDF models include the work of McAllister et al. [MLH02] which fits Lafortune lobes [LFTG97] to each pixel. While this technique achieves fairly good results for materials with low depth variation like lacquered wood, it fails to reproduce more complex BTFs due to numerical and practical problems prohibiting the use of a large number of lobes per pixel and the basic limitations of the Lafortune model. Nevertheless, the method achieves impressive compression rates of about 1:1000 and can render at real-time frame rates illuminating with either point light sources or image based lighting. An improved method was presented by Daubert et al. [DLHS01] which utilizes an additional view-dependent shadowing factor to model occlusion effects. The increased realism of the method results in worse yet still significant compression rates and can be rendered in real time using graphics hardware. Even more realistic results were generated by the approach of Meseth et al. [MMK03a] which is based on view-dependent quasi-Lafortune lobes. Unfortunately the compression rate of this method is rather low and therefore requires additional use of quantization methods. Nevertheless, it can be rendered at real-time frame rates. The technique was extended to image-based lighting in a following publication [MMK04a]. A technique achieving even better quality was published by Filip and Haindl [FH04]. While they achieve compression rates of about 1:20, their method is not well suited for real-time rendering.

3.2. Linear Basis Decomposition

BTF models belonging to the category of linear basis decomposition models are based on factorization of per pixel

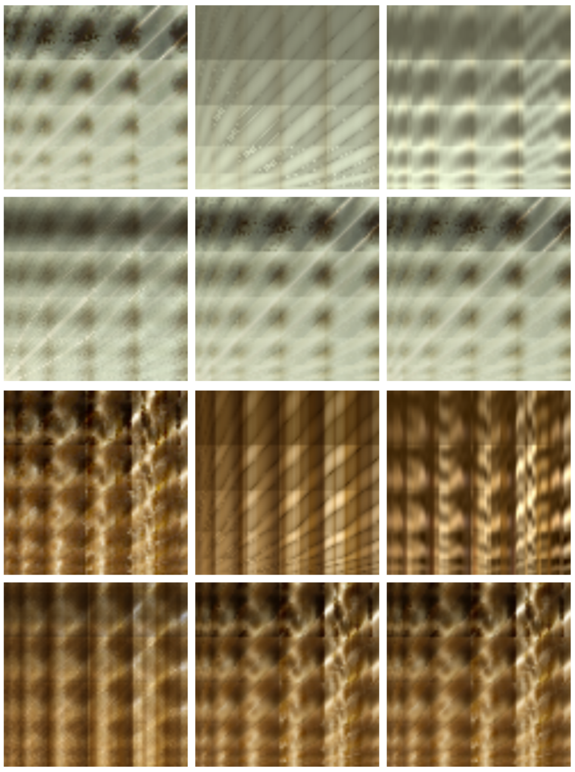


Figure 9: *The pictures compare the approximation quality of two apparent BRDFs from plaster (top) and corduroy (bottom) reconstructed from the original data (top left) to the methods of Daubert et al. [DLHS01] with two lobes (top middle), Meseth et al. [MMK04a] with a view-dependent polynomial (top right), Suykens et al. [SvBLD03] with four factors without clustering (bottom left), Sattler et al. [SSK03] with eight components (bottom middle) and Müller et al. [MMK03b] with 32 clusters and eight components (bottom right). Storage requirements for a* 256×256 *pixel BTF total to 1.2 GB, 30 MB, 106 MB, 60 MB, 121 MB and 10 MB for the respective methods.*

BRDFs and therefore assume BRDFs to be separable. Kautz and McCool [KM99] approximated the 4D $BRDF(\mathbf{v},\mathbf{l})$ by a sum of products of twodimensional functions $F_i(\mathbf{v})$ and $G_i(\mathbf{l})$ using matrix factorization. At run-time, BRDF values can efficiently be reconstructed from twodimensional textures (into which the functions were sampled) using graphics hardware. Given a reasonable number i of functions, significant compression ratios can be achieved yet the approximation yields good results for special BRDFs only. McCool et al. [MAA01] improved on the above scheme by employing homomorphic factorization, which allows multiplicative combinations or more than two functions, avoids signed arithmetic and improves the relative root mean square error (RMS) – which was found to be perceptionally more important than the absolute RMS – compared to the previous approach. A further optimization was proposed by Suykens et al. [SvBLD03]. They combine homomorphic and matrix factorization factorization to achieve good approximation results for a much larger set of BRDFs at the expense of worse data compression ratios. A comparison of the approximation quality achieved by the various BTF compression techniques 9 shows that the technique of Suykens et al. achieves acceptable results at acceptable compression rations but remains inferior to other methods. To make their scheme usable, additional clustering is applied to the compressed data resulting in visible artefacts for complex materials.

Similar to the above approaches, Koudelka et al. [KMBK03] and Liu et al. [LHZ*04] apply singular value decomposition to large matrices containing all spatially varying apparent BRDFs from the BTF to compute Eigen-BRDFs and per-pixel reconstruction weights. Unfortunately, following their schemes, high-quality approximations of general BTFs require too many coefficients for real-time rendering.

3.3. Non-Linear Basis Decomposition

A very common technique for data-compression is the PCA, which provides the in a least-squares sense optimal affine-linear approximation and which has been used by Sattler et al. [SSK03]. Basri and Jacobs [BJ03] have shown recently, that the space of images of a lambertian object under arbitrary illumination lies close to a linear 9D-subspace, which supports the hope, that a mainly diffuse and not too complex sample could be adequately reconstructed using only few components. But unfortunately, general BTFs certainly exhibit highly non-linear effects like specularities or self-occlusions due to viewpoint change which may require many coefficients in order to be reproduced in a visually satisfying quality.

Nevertheless, while being not globally linear, many high-dimensional data sets show a local linear behavior. This is the idea behind the local PCA method, which was introduced by Kambhatla and Leen [KL97] to the machine-learning community in competition to classical non-linear/neural-network learning algorithms. The method can be summarized as follows:

1. Initialize k cluster-centers r_j randomly chosen from the data set. Assign a collection of c unity basis-vectors $e_{i,j}$ to each cluster.
2. Partition the data set into regions by assigning each data-vector to its closest center. The distance to a center r_j is given by the squared reconstruction error:
$$||x-\tilde{x}||^2 = ||x-r_j-\sum_{i=1}^{c}\langle x-r_j, e_{i,j}\rangle e_{i,j}||^2$$
3. Compute new centers r_j as the mean of the data in the region j.
4. Compute a new set of basis-vectors $e_{i,j}$ per region, i.e. perform a PCA in each region.
5. Iterate steps 2.-4. until the change in average reconstruction error falls below a given threshold.

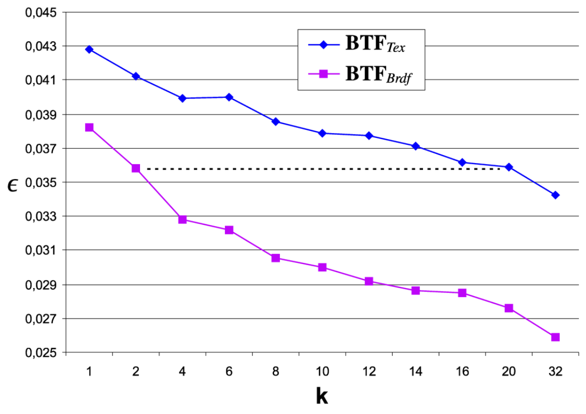


Figure 10: *Average reconstruction error for increasing number of clusters k and fixed number of components $c = 8$ (*proposte*).*

Based on the local PCA, Müller et al. [MMK03b] proposed a very efficient BTF compression scheme. To apply local PCA to the sets $\mathbf{BTF}_{Tex}$ and $\mathbf{BTF}_{Brdf}$, the elements $T_{(\mathbf{v},\mathbf{l})}$ and $B_{(x,y)}$ are simply interpreted as vectors with 3x256x256 and 3x81x81 elements and clustered into k clusters. After subtracting the respective cluster mean (denoted by $\bar{T}_k, \bar{B}_k$ respectively) from each vector, they are put into large matrices A_k. Note that the concrete implementation differs from the general idea by using only parts of the complete data as training set, because otherwise the data would exceed the maximum memory block size. The principal components for each cluster are the eigenvectors of the symmetric matrices $A_k A_k^T$. The described implementation uses the numerical package LAPACK to solve the resulting large eigen-problems. The principal components E_i^{Tex} of $\mathbf{BTF}_{Tex}$ are well known as Eigen-Textures, thus the components E_i^{Brdf} are simply called *Eigen-Brdfs*.

The reconstructions $\tilde{T}^c_{(\mathbf{v},\mathbf{l})}$ and $\tilde{B}^c_{(x,y)}$ are given as follows:

$$\begin{aligned}\tilde{T}^c_{(\mathbf{v},\mathbf{l})} &= \bar{T}+\sum_{i=1}^{c}\langle T_{(\mathbf{v},\mathbf{l})}-\bar{T}, E_i^{Tex}\rangle * E_i^{Tex}\\ \tilde{B}^c_{(x,y)} &= \bar{B}+\sum_{i=1}^{c}\langle B_{(x,y)}-\bar{B}, E_i^{Brdf}\rangle * E_i^{Brdf}\end{aligned}$$

The method was applied to both $\mathbf{BTF}_{Tex}$ and $\mathbf{BTF}_{Brdf}$ and the average absolute reconstruction error

$$\varepsilon(\mathbf{BTF}_{Tex}, c) = \sum_{(\mathbf{v},\mathbf{l})\in\mathcal{M}} \frac{||T_{(\mathbf{v},\mathbf{l})}-\tilde{T}^c_{(\mathbf{v},\mathbf{l})}||}{|I||\mathcal{M}|}$$

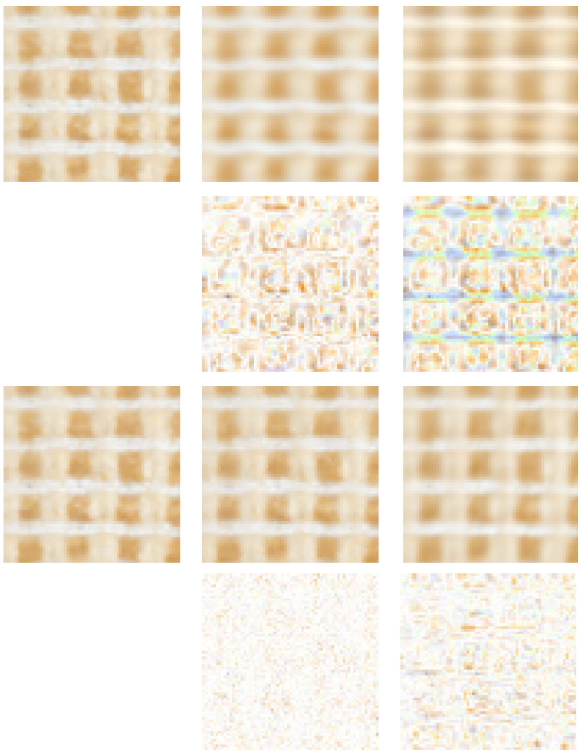


Figure 11: *The original frontal view of the* proposte *texture (left), the reconstruction for $c = 8$ without clustering (top) and with $k = 32$ clusters (bottom) from $\mathbf{BTF}_{Brdf}$ (middle) and from $\mathbf{BTF}_{Tex}$ (right) with enhanced (multiplied by 4) and inverted difference images.*

was measured, with a similar expression for $\varepsilon(\mathbf{BTF}_{Brdf}, c)$. The results of these measurements, which depend on the number of clusters k employed in reconstruction, are depicted in figure 10. One can clearly see that the apparent BRDF arrangement easily outperforms the arrangement of view-dependent textures. Note, that about twenty texture-clusters are required to achieve the approximation quality of only two BRDF-clusters. Consider also the reconstructed samples in figure 11. Similar observations were made for all measured materials and can be expected from all textures containing parts with resembling reflectance behaviour.

The reason for this observation is the increased variance for the view-dependent texture arrangement which is due to two reasons: first the variance in $\mathbf{BTF}_{Brdf}$ depends only on the complexity of the surface i.e. on *spatial* variation of reflectance properties and shadowing- and masking-effects while in $\mathbf{BTF}_{Tex}$ additional variance is introduced by the measurement process i.e. registration-errors and filtering. Second, specularities introduce strong variance among the textures, while their variation across the BRDFs depends only on material structure.

Another great advantage of the BRDF-wise arrangement arises from the fact that the spatial sampling density is usually higher than the angular one ($|I| > |\mathcal{M}|$). The memory needed for an additional cluster is given by the number of the used principal components which have dimension $|I|$ or $|\mathcal{M}|$ respectively. In the above case $|I| \approx 10 * |\mathcal{M}|$ what is also reflected in table 2. In a typical case with $k = 32, c = 8$ a compression ratio of about 1:100 against 1:10 is achieved. In this context, the BRDF-wise approach is especially suited for materials that contain both high- and low-frequency surface structure and therefore require a high spatial sampling frequency.

Since the BRDF-wise arrangement outperforms the arrangement as view-dependent textures, the remaining text will always assume the first data arrangement and therefore will assume the index Brdf of $\mathbf{BTF}_{Brdf}$ and E^{Brdf}.

	$k\backslash c$	4	6	8	10
Tex	2	3.2	4.9	6.4	8
Brdf		0.84	1.25	1.68	2.09
	4	6.3	9.5	12.7	15.9
		1.15	1.73	2.3	2.9
	8	12.6	18.9	25.3	31.6
		1.78	2.68	3.57	4.46
	16	25.2	37.8	50.4	63
		3.04	4.6	6	7.6
	32	50.4	75.5	100.8	125.9
		5.6	8.4	11.1	13.9

Table 2: *Size of the local PCA-compressed data in* million bytes *depending on the number of clusters k and components c. The data format is 16-bit float.*

4. Real-Time Rendering

In general accurate rendering algorithms have to compute results approximating the rendering equation for every surface point $\mathbf{x}$ by computing outgoing radiance L_r as follows (neglecting emissive effects):

$$L_r(\mathbf{x}, \mathbf{v}) = \int_{\Omega_i} BRDF_{\mathbf{x}}(\mathbf{v}, \mathbf{l}) L_i(\mathbf{x}, \mathbf{l}) (\mathbf{n}_x \cdot \mathbf{l})\, d\mathbf{l}$$

Here, $BRDF_{\mathbf{x}}$ denotes the BRDF for point $\mathbf{x}$, L_i denotes incoming radiance, $\mathbf{n}_x$ is the surface normal and Ω_i is the hemisphere over $\mathbf{x}$. Employing measured BTFs, the following approximation results:

$$L_r(\mathbf{x},\mathbf{v}) \approx \int_{\Omega_i} \mathbf{BTF}(\mathbf{x},\mathbf{v},\mathbf{l}) L_i(\mathbf{x},\mathbf{l})\, d\mathbf{l} \qquad (1)$$

The area foreshortening term is removed since this effect is represented in the measured BTFs already.

4.1. Point Light Source Illumination

In the presence of a finite number of point light sources only, the integral reduces to a sum. Approximating the BTF by the clustered Eigen-BRDFs, the above equation reduces to

$$L_r(\mathbf{x},\mathbf{v}) \approx \sum_{j=1}^{n}\sum_{l=1}^{c} \alpha_{l,k(\mathbf{x})} E_{l,k(\mathbf{x})}(\mathbf{v},\mathbf{l}_j) L_i(\mathbf{x},\mathbf{l}_j)$$

Here, n denotes the number of light sources, α denotes the projections on the respective basis vectors as in section 3.3, c is the number of components from the clustered Eigen-BRDFs $E_{k(\mathbf{x}),l}$ and $k(\mathbf{x})$ is the cluster or material index. Since the BTF was sampled for fixed light and view directions only, linear interpolation is employed to support arbitrary view and light directions, resulting in the final equation which has to be evaluated on graphics hardware:

$$L_r(\mathbf{x},\mathbf{v}) \approx \sum_{j=1}^{n} \sum_{\substack{\tilde{\mathbf{v}}\in N(\mathbf{v})\\ \tilde{\mathbf{l}}\in N(\mathbf{l}_j)}} w_{\tilde{\mathbf{v}},\tilde{\mathbf{l}}} \sum_{l=1}^{c} \alpha_{l,k(\mathbf{x})} E_{l,k(\mathbf{x})}(\tilde{\mathbf{v}},\tilde{\mathbf{l}}) L_i(\mathbf{x},\mathbf{l}_j)$$

By $N(\mathbf{v})$ we denote the set of neighboring view directions of $\mathbf{v}$, for which data was measured ($N(\mathbf{l}_j)$ respectively), while w denotes an interpolation weight.

Rendering algorithms for this approach have to optimize performance by optimizing the time for reconstructing BTF values and optimizing the overhead for interpolation of the closest BTF samples. Both goals can efficiently be reached using graphics hardware - to be precise: employing fragment programs and hardware supported filtering - although special efforts have to be made to achieve the second goal using current graphics hardware which usually does not support quadri-linear filtering. In the following text, first efficient 4D interpolation will be addressed and then an efficient fragment shader implementation will be presented.

4.1.1. Efficient Interpolation

The presented implementation from Schneider [Sch04] employs an approach similar in style to Liu et al. [LHZ*04]: the tri-linear filtering capabilities of graphics hardware are used to interpolate the view direction and the polar angle of the light direction. The final step for 4D filtering is performed manually by blending two, tri-linearly filtered values with closest azimuth angle in light direction.

The crucial decision for achieving accurate results for 4D filtering using 3D filtering hardware is parameterization of the data to be filtered. Since our data is sampled from directions approximately equally distributed over the hemisphere we have to find a mapping from the hemisphere to a rectangular grid which preserves this uniform sampling i.e. which ensures that the viewing and lighting directions corresponding to a uniform sampling grid of the parameter space are evenly distributed on the hemisphere.

Such a mapping, which is commonly employed in computer graphics, is parabolic mapping [HS98]. Directions $(r_x,r_y,r_z)^t = (cos\phi sin\theta, sin\phi sin\theta, cos\theta)^t$ with $\theta \in [0,\pi]$ and $\phi \in [0,2\pi]$ are mapped to points $(s,t) \in [0,1]^2$ on the rectangular surface as follows:

$$\begin{pmatrix} s \\ t \end{pmatrix} = -\frac{0.5}{r_z+1}\begin{pmatrix} r_x \\ r_y \end{pmatrix} + \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} \qquad (2)$$

Given this mapping, the Eigen-BRDFs can be reparameterized as follows:

$$\mathbf{E}_{i,k(x)}(\theta_v,\phi_v,\theta_l,\phi_l) = \tilde{\mathbf{E}}_{i,k(x)}(s_v,t_v,s_l,t_l)$$

This reparamaterized representation can be used for hardware accelerated rendering by discretizing the lighting parameter t_l as $\{t_{l_1},\dots,t_{l_n}\}$. A set of functions with fixed t_l for all n provided values of t_l is created and stored in a 3D texture

$$S_i(s_v,t_v,s_l) = \tilde{E}_{j,k(x)}(s_v,t_v,s_l,t_{l_i}).$$

Now the interpolation in the s_v,t_v and s_l dimension are performed by the hardware and the fragment shader has to handle only the interpolation in the fourth dimension t_l by itself. This is done by performing two texture accesses and linearly interpolating the results afterwards.

$$\tilde{E}_{j,k(x)}(s_v,t_v,s_l,t_l) = \begin{cases} S_0(s_v,t_v,s_l) & ,t_l < t_{l_0} \\ S_{n(t_l)-1}(s_v,t_v,s_l) & ,t_l \geq t_{l_{n(t_l)-1}} \\ (1-w)S_i(s_v,t_v,s_l) & \\ +wS_{i+1}(s_v,t_v,s_l) & ,t_{l_i} \leq t_l < t_{l_{i+1}} \end{cases}$$

with $w = (t_l - t_{l_i})/(t_{l_{i+1}} - t_{l_i})$.

In practice the sequence $S_i(s_v,t_v,s_l)$ of volume textures is combined into a single volume texture and loaded into graphics hardware. The texture is indexed by the texture coordinates

$$(s_v,t_v,z_1 = t_{l_i} + \tfrac{s_l}{n(t_l)})$$
$$(s_v,t_v,z_2 = t_{l_{i+1}} + \tfrac{s_l}{n(t_l)})$$

The cost of an Eigen-BRDF interpolation is therefore two texture accesses and one linear interpolation.

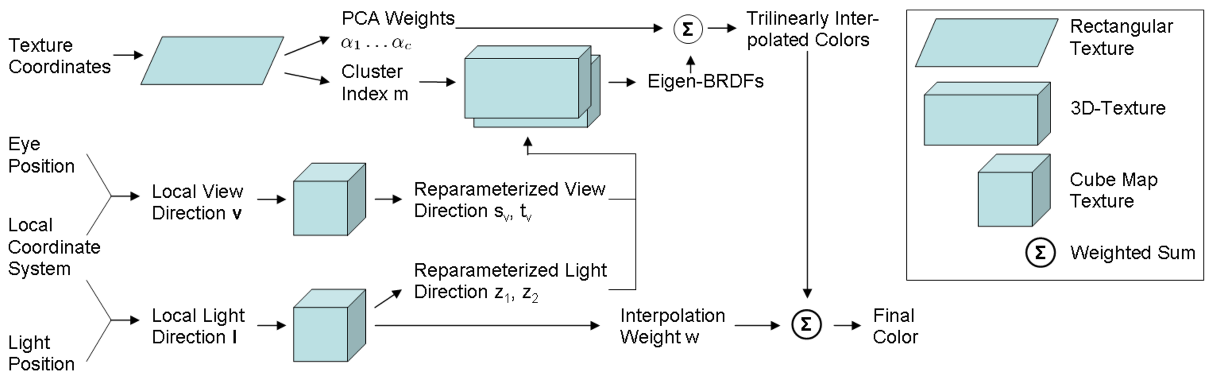

Texture Coordinates
PCA Weights
$\alpha_1 \dots \alpha_c$
Cluster Index m
Eigen-BRDFs
Trilinearly Interpolated Colors
Eye Position
Local Coordinate System
Light Position
Local View Direction **v**
Reparameterized View Direction s_v, t_v
Local Light Direction **l**
Reparameterized Light Direction z_1, z_2
Interpolation Weight w
Final Color
Rectangular Texture
3D-Texture
Cube Map Texture
Weighted Sum

Figure 12: *Layout of the fragment shader for LPCA-rendering. Interpolation of t_l is done manually.*

4.1.2. Efficient BRDF Evaluation

The remaining data required for reconstruction of the BTF is stored in another texture as depicted in figure 12. The rectangular texture stores material cluster indices together with sets of floating-point PCA weights α which determine, how the components of the indexed Eigen-BRDFs are to be combined.

In order to avoid the fragment shader's online effort of calculating the viewing parameters (s_v, t_v) from (θ_v, ϕ_v) according to equation 2 the mapping is precomputed and stored in a texture. Therefore the hemisphere of directions is sampled densely and for every direction the corresponding (s_v, t_v) is stored in a cube map. In addition to (s_v, t_v) also z_1, z_2 and w values are precomputed and put into a second cube map.

Another issue is dealing with the floating point representation of the Eigen-BRDFs. Since nVIDIA's GeForce FX only supports 3D textures with 8-bit precision every Eigen-BRDF is quantized to 8-bit separately yielding scaling factors $s_{j,k}$. To compensate for this every floating-point PCA weight $\alpha_{j,k}$ is divided by the corresponding scaling factor $s_{j,k}$ in order to avoid performing the rescaling at runtime. As a result, two different 3D textures (one for the mean and one for the components) are used.

The rendering process as encoded in the fragment shader is depicted in figure 12. The fragment program's inputs are standard texture coordinates, the eye and light positions, and a per-pixel coordinate system, which is interpolated from the local coordinate systems at the vertices which are specified with the geometry and stored in display lists.

At first view and light directions are computed and transformed into the pixel's coordinate system. Using the cube maps, the texture coordinates (s,t) for the reparameterized Eigen-BRDFs and the interpolation weight are looked up. Next, using the texture coordinates, cluster indices and PCA weights for every pixel on the screen are looked up. The cluster index and (s,t) are combined and used to select the appropriate RGB components from the appropriate 81x81x3 vectors representing the c Eigen-BRDFs of the current cluster. Combining the PCA weights of the current pixel with the RGB components, two trilinearly interpolated colors are computed. In a final step, these colors are multiplied with their respective interpolation weights and the results are summed to form the final color of the pixel.

Mipmapping the BTF can simply be implemented by executing the shader instructions twice (once for the currently best mipmap level and once for the next best mipmap level) and interpolating the resulting colors. Bilinear, spatial interpolation is currently not supported since the additional overhead is prohibitiv. Fortunately, hardware supported full-screen antialiasing can reduce potential artefacts significantly.

The presented rendering method was implemented on a NVidia Geforce FX 5950 graphics board and tested with several models and BTF-data from the BTF Database Bonn [BTF]. For moderate projection sizes of the models of about 640×480 and a limited number of PCA components (up to four) the models render with interactive to real-time frame rates well over 10 fps which is sufficient for most cloth rendering applications but has to be improved to be useful e.g. in computer games. Examples of rendered images are shown in figure 13

4.2. Image-Based Lighting and Shadows

The approach of lighting scenes with point or directional light sources is very widely spread in computer graphics and especially Virtual Reality (VR) applications due to its simplicity and the existing support in graphics hardware and graphics APIs. Despite this frequent use, point and directional light source illumination tends to appear synthetic since real-world light sources usually differ significantly from these simplified types of lights. In reality, large fractions of the incoming radiance result from indirect lighting e.g. from walls in buildings or clouds and the air in outdoor scenes. Since real-time cloth visualization tries to achieve

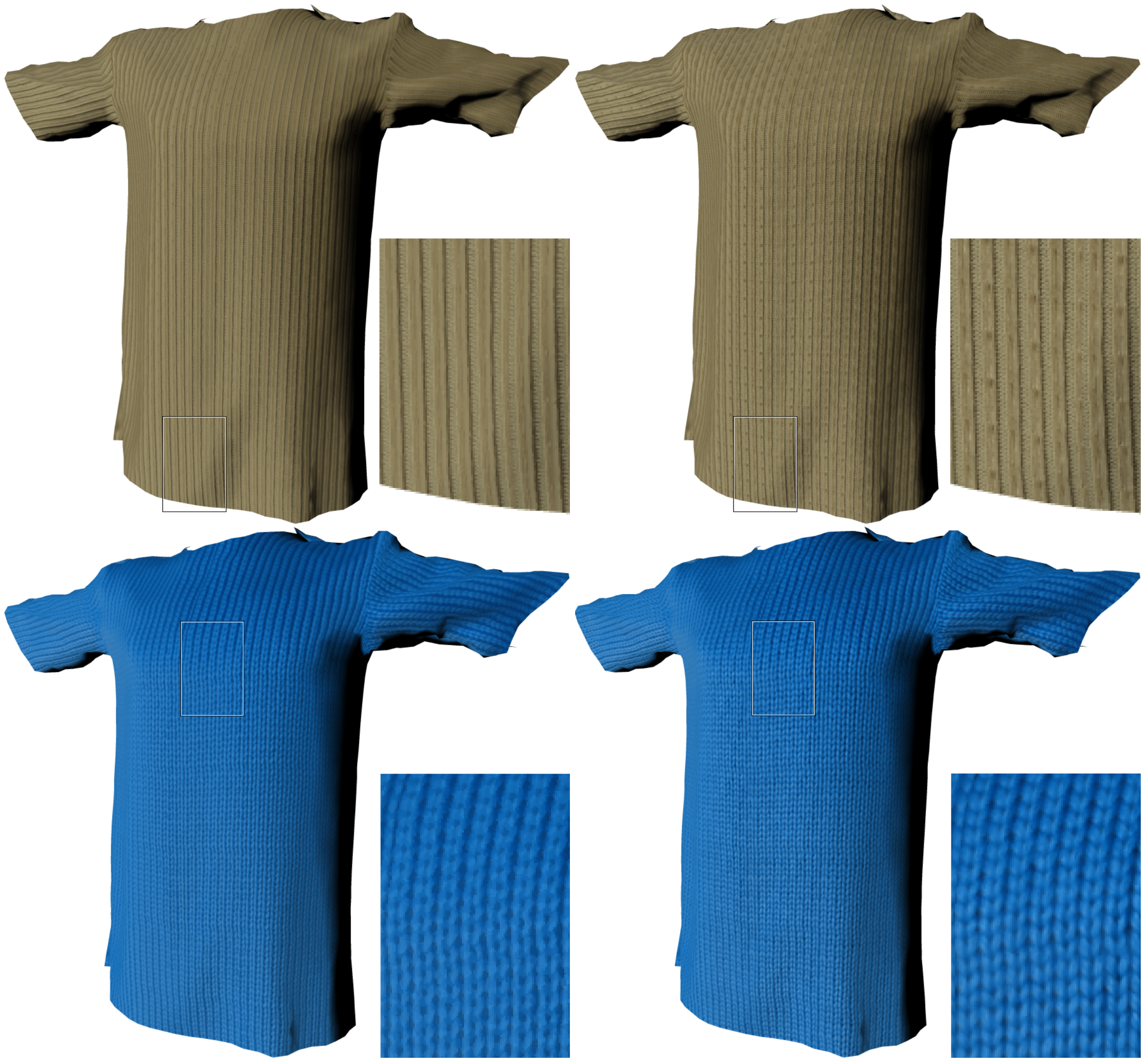


Figure 13: *Results from the local PCA rendering algorithm for varying numbers of PCA components. Top left: mean only, top right: mean and 4 components, bottom left: mean only, bottom right: mean and 2 components. While the rendered detail increases significantly in the top row pictures as visible in the closeups, the quality of the wool improves only slightly.*

as realistic as possible results, integration of such complex lighting scenarios represents an important goal of rendering algorithms.

Correctly simulating the light interchange of objects in the scene constitutes a partially not understood but definitely run-time intense task which can usually not be computed in real-time. Existing real-time applications therefore make simplifying assumptions like omission of interreflection between the lit objects and the remaining scene and very distant lighting environments. Many approaches further neglect self-shadowing and self-interreflections of the lit object and additionally assume either fixed, spatially varying BRDF types [HS99, MLH02, MMK04a], fixed type [KVHS00] or general isotropic [KS00, LK02] BRDFs. Others include self-shadowing and self-interreflections, allow interactive changes of the lighting environment but assume low-frequency lighting and spatially fixed [RH01, RH02, SKS02, LK03, SHHS03] or spatially varying BRDFs [SLSS03, MMK04b], or do not achieve real-time frame rates yet [NRH03, NRH04].

4.2.1. Illumination and Material Representation

In the remaining text the approach of Müller et al. [MMK04b] will be presented since it applies to BTFs and especially local PCA encoded BTFs which were presented in the previous section. The method assumes a very distant illumination environment which allows to drop the spatial dependence of the incoming light in equation 1. In addition, low frequency lighting is assumed which can efficiently be represented in a low-order Spherical Harmonics basis expansion (a very comprehensible introduction to spherical harmonics is given in [SKS02]). Representing incoming radiance L_i and BTF for each texel $\mathbf{x}$ and view direction $\mathbf{v}$ in an n-th order (typically n=5) SH-basis expansion yields:

$$\begin{aligned} L_i(\mathbf{l}) &\approx \sum_j^{n^2} a_j y_j(\mathbf{l}) \\ \mathbf{BTF}(\mathbf{x},\mathbf{v},\mathbf{l}) &\approx \sum_j^{n^2} b_j^{\mathbf{x},\mathbf{v}} y_j(\mathbf{l}) \end{aligned}$$

where y_j denote the Spherical Harmonics basis functions and the projection coefficients a_j and $b_j^{\mathbf{x},\mathbf{v}}$ can be obtained via numerical integration. Due to the ortho-normality of the SH basis the lighting-integral now reduces to a simple dot product:

$$L_r(\mathbf{x},\mathbf{v}) = \langle \mathbf{b}^{\mathbf{x},\mathbf{v}}, \mathbf{M_x}(\mathbf{a}_i) \rangle \tag{3}$$

Here $\mathbf{a}_i)$ and $\mathbf{b}^{\mathbf{x},\mathbf{v}}$ denote the vectors of stacked SH coefficients and $\mathbf{M_x}$ denotes a high-dimensional transfer matrix [SKS02] which rotates the lighting environment into the local coordinate frame and additionally encodes for example large scale shadows and inter-reflections. Transfer matrices can efficiently be computed using Monte-Carlo light simulation. To combine low-frequency image-based lighting with local PCA encoded BTFs, $\mathbf{M_x}$ is expanded into a local PCA basis as well:

$$\mathbf{M_x} \approx \sum_{j=0}^{c_M} \alpha_{\mathbf{x},j} \mathbf{m}_{j,k_M(\mathbf{x})}.$$

The view-dependent SH-coefficients for every BTF-texel can be reconstructed from the mean and Eigen-BRDF SH-coefficients $\mathbf{e}^{\mathbf{v}}_{j,k_M(\mathbf{x})}$:

$$\mathbf{b}^{\mathbf{v},\mathbf{x}} \approx \sum_{j=0}^{c_B} \cdot \mathbf{e}^{\mathbf{v}}_{j,k_M(\mathbf{x})}$$

The cluster index look-up is denoted with $k_{M()}$ for the Matrix clusters and $k_{B()}$ for the BTF clusters. Now equation 3 can be expressed in terms of the LPCA components:

$$L_r(\mathbf{x},\mathbf{v}) \approx \sum_{j=0}^{c_M} \sum_{m=0}^{c_B} \alpha_{\mathbf{x},j} \beta_{\mathbf{x},m} \left\langle \mathbf{e}^{\mathbf{v}}_{m,k_M(\mathbf{x}}, \mathbf{m}_{j,k_M(\mathbf{x})}(\mathbf{a}_i) \right\rangle \tag{4}$$

The equation reduces the computation of the lighting integral for a fixed lighting environment to a simple weighted sum of dot products that can be precomputed independently from screen resolution and mesh-complexity. The dot products

$$\lambda^{\mathbf{v}}_{m,k_B,j,k_M}(\mathbf{a}_i) = \langle \mathbf{e}_{m,k_M}, \mathbf{m}_{j,k_M}(\mathbf{a}_i) \rangle$$

must be recomputed only if the lighting vector $\mathbf{a}_i$ changes. Of course the cost for this computation increases with the number of clusters and components since $|k_M| \cdot (c_M + 1)$ matrix-vector multiplications and $|k_B| \cdot |k_M| \cdot (c_b + 1) \cdot (c_M + 1) \cdot |v|$ dot products have to be computed for each color channel. In cases when the rendering speed is decreased too much due to this precomputation, one can temporarily reduce the number of matrix or BTF components thereby trading quality for speed. Such situations will occur when lighting environments are rotated around the object - or equivalently if the object is turned in the fixed lighting environment. Since quality losses in situations of changing lighting are less visible to the user, reducing the quality does not impact the usability of this method significantly.

4.2.2. Real-Time Rendering

Rendering meshes covered with LPCA encoded BTFs in environments with arbitrary lighting (represented as environment maps) consists of two steps: precomputing the parameters for the current lighting situation on the CPU and rendering the mesh using these parameters.

To efficiently access the precomputed PRT values

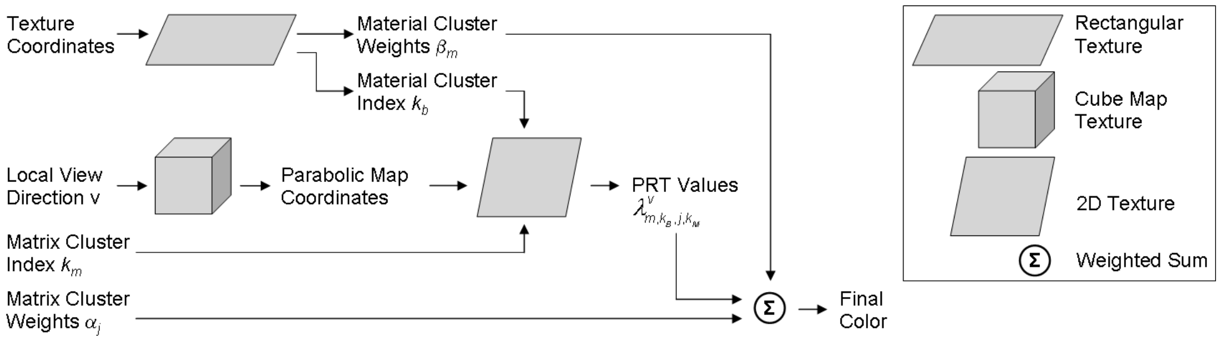

Texture Coordinates
Material Cluster Weights β_m
Material Cluster Index k_b
Local View Direction v
Parabolic Map Coordinates
PRT Values λ^v_{m,k_B,j,k_M}
Matrix Cluster Index k_m
Matrix Cluster Weights α_j
Σ
Final Color
Rectangular Texture
Cube Map Texture
2D Texture
Σ Weighted Sum

Figure 14: *Schematic setup of the fragment shader for image based lighting of local PCA encoded BTFs*

$\lambda^{\mathbf{v}}_{m,k_B,j,k_M}$ during hardware supported rendering they are stored in parabolic hemispherical maps [HS98]. That way, interpolation between sampled view-directions can be achieved employing the bilinear filtering capabilities of graphics hardware. In all conducted tests a resolution of 16×16 texels led to satisfying results.

Since current graphics hardware prohibits linear interpolation of floating-point valued textures (this limitation will be removed with nVIDIA's NV40 chip), one has to map the high-dynamic range PRT values to 8 bit integers. Since the values differ significantly but can roughly be categorized into four categories (one category for $m = j = 0$, a second for $m = 0$ and $j \neq 0$, a third for $m \neq 0$ and $j = 0$, and a fourth for $m \neq 0$ and $j \neq 0$), four different scaling values are computed, applied to the PRT values before storage in a texture and the inverse factors are sent to the graphics board.

During rendering, special attention has to be payed to the way the models are rendered due to the matrix clusters which are assigned to vertices only, other than the BTF clusters which are assigned per pixel. Therefore one has to make sure that in each rendering pass only triangles belonging to a single matrix cluster are rendered and that the rendering results for triangles whose vertices belong to different clusters are blended appropriately to achieve smoothly changing lighting over the triangle.

Therefore, a setup similar to Sloan et al. [SHHS03] is used. The triangles of the mesh are sorted into bins according to the matrix clusters of the vertices of the triangles (i.e. whenever at least one vertex of the triangle belongs to the matrix cluster assigned to the bin, the triangle is inserted into the bin). For every triangle in every bin three Boolean parameters indicating whether the vertices belong to the bin's cluster are stored. At run-time the triangles in the bins are rendered independently. Taking into account the Boolean parameters, only vertices belonging to the bin's rotation cluster contribute to the partial image produced by the bin. The partial results are accumulated by setting the blending mode to add.

Other than in Sloan et al. [SHHS03], the functionality of the employed vertex shader is limited to computing the local view direction and setting the matrix cluster weights to zero if the Boolean parameter of the vertex is set to false. The majority of computation is performed in the fragment shaders (see figure 14 for a schematic description). The inputs to the shader are the texture coordinates of the current fragment, the local view direction (which is interpolated from the local view directions at the vertices), the matrix cluster index and the interpolated matrix cluster weights. Based on the texture coordinates the material cluster index and the material cluster weights are lookup up. The local view direction serves as index into a cube map that returns corresponding Parabolic Map coordinates. Based on these coordinates, the matrix and material cluster indices, PRT values are looked up from a 2D texture - one for each material and matrix cluster. The final color of the fragment is computed according to equation 3. Please note that figure 14 omits the necessary scaling of the 8 bit PRT values.

The described rendering algorithm was implemented on an Intel Pentium IV 2.6 GHz processor with a GeForce FX 5900 graphics board using OpenGL.

Precomputing the PRT values whenever the light situation changes takes about 0.8 seconds for 32 matrix clusters, 8 matrix components, 16 BTF clusters and 4 BTF components - a high-quality setting that suffices for the tested models and materials. Since this number is much too high for interactive light changes, a smaller numbers of components is chosen whenever the model is rotated in the environment (i.e. only 4 matrix and 1 BTF components). Since the preprocessing time decreases approximately linearly with the number of clusters and components, the precomputation time is reduced so much that even real-time navigation becomes possible. Although not implemented in the framework, this technique naturally applies to varying lighting environments (e.g. due to video textures) as well.

Rendering times for the meshes mainly depend on the screen projection size of the model and on the number of components (both matrix and BTF) employed. Models pre-

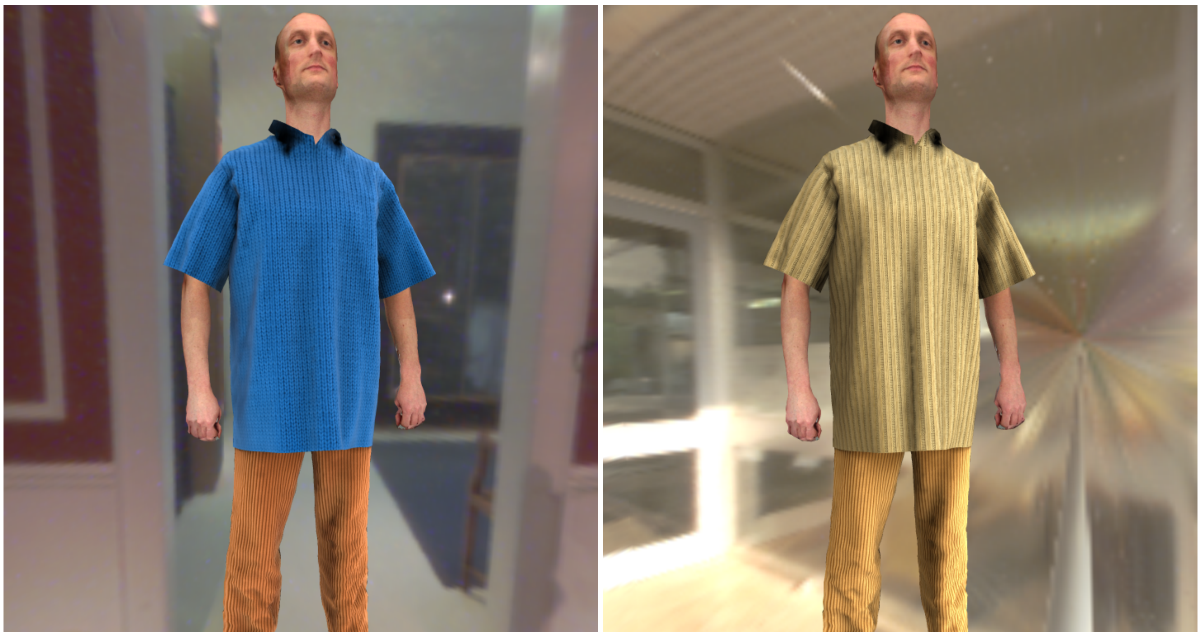


Figure 15: *Avatar wearing different cloths in the Shop (left) and UBO (right) environment.*

sented in this tutorial section render at close to real-time frame rates (about 12 fps) even at high resolutions. These frame-rates can as well be achieved for models covered with multiple BTFs.

Figure 15 shows pictures of an avatar wearing clothes made of different materials stored as LPCA encoded BTFs which are lit by image-based lighting. The large-scale shadows are clearly visible. Please note the different color tones in the images resulting from the differently colored lighting environments. Figure 16 compares the appearance of the Max Planck bust lit by several different lighting environments and demonstrates the effect of large-scale shadowing.

5. Conclusions

In this tutorial, the necessity of realistic material representations for realistic cloth visualization was demonstrated. Methods and algorithms for acquisition, compression and real-time rendering of such realistic materials represented as BTFs were presented and compared, and results of these approaches were shown.

Acknowledgements

Many results presented in this tutorial were at least partially funded by the German Ministry of Education and Research (BMB+F) under project Virtual Try-On (01IRA01A) and the European Union under project RealReflect (IST-2001-34744). Some of the HDR environments in figure 16 were kindly provided by Paul Debevec.

References

[Ame02] AMERICAN NATIONAL STANDARDS INSTITUTE: *Photography - Electronic still picture imaging - Reference Input Medium Metric RGB Color encoding (RIMM-RGB)*, 2002. Document # ANSI/I3A IT10.7466-2002. 4

[BJ03] BASRI R., JACOBS D. W.: Lambertian Reflectance and Linear Subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*, 2 (2003), 218–233. 7

[BTF] BTF Database Bonn. http://btf.cs.uni-bonn.de/. 5, 10

[CUR] CUReT: http://www.cs.columbia.edu/CAVE/curet/. 2

[DLHS01] DAUBERT K., LENSCH H., HEIDRICH W., SEIDEL H.-P.: Efficient Cloth Modeling and Rendering. In *12th Eurographics Workshop on Rendering* (2001), pp. 63–70. 3, 6

[DS02] DAUBERT K., SEIDEL H.-P.: Hardware-Based Volumetric Knit-Wear. *Computer Graphics Forum 21*, 3 (2002), 314–325. 3

[DvGNK97] DANA K. J., VAN GINNEKEN B., NAYRA S. K., KOENDERINK J. J.: Reflectance and Texture of Real World Surfaces. In *IEEE Conference on Computer Vision and Pattern Recognition* (San Juan, Puerto Rico, June 1997), pp. 151–157. 2

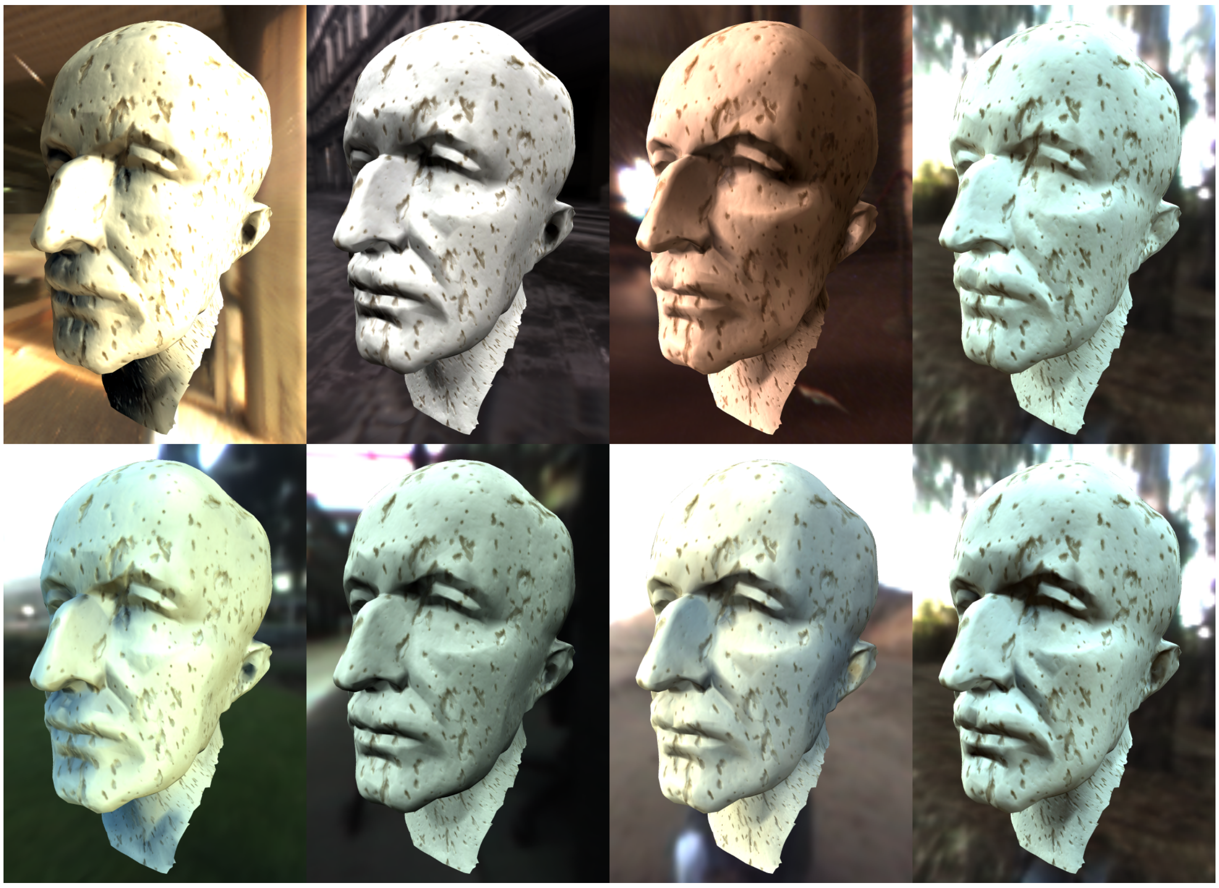


Figure 16: *Max Planck bust covered by Plasterstone lit by lighting environments. From left to right, bottom to top: UBO, Uffizi, Galileo, RNL, Campus, Building, Beach, RNL. All images but the rightmost on the top are rendered with large-scale shadows.*

[DvGNK99] DANA K. J., VAN GINNEKEN B., NAYAR S. K., KOENDERINK J. J.: Reflectance and Texture of Real-World Surfaces. *ACM Transactions on Graphics 18*, 1 (1999), 1–34. 4

[FH04] FILIP J., HAINDL M.: Non-linear Reflectance Model for Bidirectional Texture Function Synthesis. In *ICPR 2004* (2004). 6

[GRS96] GRÖLLER E., RAU R., STRASSER W.: Modeling Textiles as Three Dimensional Textures. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96* (1996). 3

[HEE*02] HAUTH M., ETZMUSS O., EBERHARDT B., KLEIN R., SARLETTE R., SATTLER M., DAUBER K., KAUTZ J.: Cloth Animation and Rendering. In *Eurographics 2002 Tutorials* (2002). 3

[HS98] HEIDRICH W., SEIDEL H.-P.: View-Independent Environment Maps. In *Proceedings of Eurographics/SIGGRAPH Workshop on Graphics Hardware '98* (1998), pp. 39–45. 9, 13

[HS99] HEIDRICH W., SEIDEL H.-P.: Realistic, Hardware-Accelerated Shading and Lighting. In *SIGGRAPH '99* (Los Angeles, CA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 171–178. ISBN:0-201-48560-5. 12

[KL97] KAMBHATLA N., LEEN T.: Dimension Reduction by Local Principal Component Analysis. *Neural Computation 9* (1997), 1493–1516. 7

[KM99] KAUTZ J., MCCOOL M.: Interactive Rendering with Arbitrary BRDFs using Separable Approximations. In *Tenth Eurographics Workshop on Rendering* (1999), pp. 281–292. 6

[KMBK03] KOUDELKA M. L., MAGDA S., BELHUMEUR P. N., KRIEGMAN D. L.: Acquisition, Compression, and Synthesis of Bidirectional Texture Functions. In *3rd International Workshop on Texture Analysis and Synthesis* (2003). 7

[KS00] KAUTZ J., SEIDEL H.-P.: Towards Interactive Bump Mapping with Anisotropic Shift-Variant BRDFs. In *2000 SIGGRAPH/EUROGRAPHICS Workshop On Graphics Hardware* (Interlaken, Switzerland, 2000), ACM Press, pp. 51–58. 12

[KVHS00] KAUTZ J., VÁZQUEZ P.-P., HEIDRICH W., SEIDEL H.-P.: A Unified Approach to Prefiltered Environment Maps. In *11th Eurographics Workshop on Rendering* (2000), pp. 185–196. 12

[LFTG97] LAFORTUNE E. P. F., FOO S.-C., TORRANCE K. E., GREENBERG D. P.: Non-linear Approximation of Reflectance Functions. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (1997), ACM Press/Addison-Wesley Publishing Co., pp. 117–126. 6

[LHZ*04] LIU X., HU Y., ZHANG J., TONG X., GUO B., SHUM H.-Y.: Synthesis and Rendering of Bidirectional Texture Functions on Arbitrary Surfaces. *IEEE Transactions on Visualization and Computer Graphics 10*, 3 (2004), 278–289. 7, 9

[LK02] LATTA L., KOLB A.: Homomorphic Factorization of BRDF-Based Lighting Computation. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), ACM Press, pp. 509–516. 12

[LK03] LEHTINEN J., KAUTZ J.: Matrix Radiance Transfer. In *Symposium on Interactive 3D Graphics 2003* (April 2003), pp. 59–64. 12

[LLSS03] LENSCH H., LANG J., SÁ A. M., SEIDEL H.: Planned Sampling of Spatially Varying BRDFs. *Computer Graphics Forum 22*, 3 (September 2003), 473–482. 4

[LYS01] LIU X., YU Y., SHUM H.-Y.: Synthesizing Bidirectional Texture Functions for Real-World Surfaces. In *Proceedings of SIGGRAPH 2001* (2001), pp. 97–106. 3

[MAA01] MCCOOL M. D., ANG J., AHMAD A.: Homomorphic Factorization of BRDFs for High-Performance Rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM Press, pp. 171–178. 6

[McA02] MCALLISTER D. K.: *A Generalized Surface Appearance Representation for Computer Graphics*. PhD thesis, Department of Computer Science, University of North Carolina at Chapel Hill, 2002. 3

[MGW01] MALZBENDER T., GELB D., WOLTERS H.: Polynomial Texture Maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM Press, pp. 519–528. 3

[MLH02] MCALLISTER D. K., LASTRA A., HEIDRICH W.: Efficient Rendering of Spatial Bi-directional Reflectance Distribution Functions. In *Graphics Hardware 2002* (Saarbrucken, Germany, 2002), Eurographics Association, pp. 79–88. ISBN:1-58113-580-7. 6, 12

[MMK03a] MESETH J., MÜLLER G., KLEIN R.: Preserving Realism in real-time Rendering of Bidirectional Texture Functions. In *OpenSG Symposium 2003* (April 2003), Eurographics Association, Switzerland, pp. 89–96. 6

[MMK03b] MÜLLER G., MESETH J., KLEIN R.: Compression and real-time Rendering of Measured BTFs using local PCA. In *Vision, Modeling and Visualisation 2003* (November 2003), pp. 271–280. 6, 7

[MMK04a] MESETH J., MÜLLER G., KLEIN R.: Reflectance Field based real-time, high-quality Rendering of Bidirectional Texture Functions. *Computers and Graphics 28*, 1 (February 2004), 103–112. 6, 12

[MMK04b] MÜLLER G., MESETH J., KLEIN R.: Fast Environmental Lighting for Local-PCA Encoded BTFs. In *Computer Graphics International 2004 (CGI2004)* (June 2004). 12

[NRH*77] NICODEMUS F. E., RICHMOND J. C., HSIA J. J., GINSBERG I. W., LIMPERIS T.: *Geometrical Considerations and Nomenclature for Reflectance*. No. 160 in Monograph. US Department of Commerce, National Bureau of Standards, NBS, Washington, DC, 1977. 2

[NRH03] NG R., RAMAMOORTHI R., HANRAHAN P.: All-Frequency Shadows using Non-Linear Wavelet Lighting Approximation. *ACM Transactions on Graphics 22*, 3 (2003), 376–381. 12

[NRH04] NG R., RAMAMOORTHI R., HANRAHAN P.: Triple Product Wavelet Integrals for All-Frequency Relighting. to appear in Proceedings of SIGGRAPH 2004, 2004. 12

[RH01] RAMAMOORTHI R., HANRAHAN P.: An Efficient Representation for Irradiance Environment Maps. In *Proceedings of SIGGRAPH 2001* (2001), pp. 497–500. 12

[RH02] RAMAMOORTHI R., HANRAHAN P.: Frequency Space Environment Map Rendering. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), ACM Press, pp. 517–526. 12

[Rob96] ROBERT L.: Camera Calibration without Feature Extraction. *Computer Vision and Image Understanding: CVIU 63*, 2 (1996), 314–325. 3

[Sch04] SCHNEIDER M.: Real-Time BTF Rendering. In *Proceedings of CESCG 2004* (2004). 9

[SHHS03] SLOAN P.-P., HALL J., HART J., SNYDER J.: Clustered Principal Components for Precomputed Radiance Transfer. *ACM Transactions on Graphics 22*, 3 (2003), 382–391. 12, 13

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), ACM Press, pp. 527–536. 12

[SLSS03] SLOAN P.-P., LIU X., SHUM H.-Y., SNYDER J.: Bi-Scale Radiance Transfer. *ACM Transactions on Graphics 22*, 3 (2003), 370–375. 12

[SSK03] SATTLER M., SARLETTE R., KLEIN R.: Efficient and Realistic Visualization of Cloth. In *Proceedings of Eurographics Symposium on Rendering 2003* (Leuven, Belgium, June 2003). 5, 6, 7

[SvBLD03] SUYKENS F., VOM BERGE K., LAGAE A., DUTRÉ P.: Interactive Rendering of Bidirectional Texture Functions. In *Eurographics 2003* (September 2003), pp. 463–472. 6, 7

Context-Specific Cloth Simulation

(Tutorial Notes)

Frederic Cordier, Nadia Magnenat-Thalmann

MIRALab, CUI, University of Geneva, Switzerland

E-mail: {cordier, thalmann}@miralab.unige.ch

Abstract

In this chapter, we describe two methods for cloth animation in real-time. These two algorithms work in a hybrid manner exploiting the merits of both the physical-based and geometric deformations. They make use of predetermined conditions between the cloth and the body model, avoiding complex collision detection and physical deformations wherever possible. Garments are segmented into pieces that are simulated by various algorithms, depending on how they are laid on the body surface and whether they stick or flow on it. Tests show that these methods are well suited to fully-dressed virtual human models, achieving real-time performance compared to ordinary cloth-simulations.

1 Introduction

One of the most challenging areas in research is in the development of a robust methodology for simulating clothes in real-time. In order to define a cloth simulation system that is able to simulate complex garments realistically, whilst maintaining a reasonable computation time, a deeper study of the cloth model and the identification of its behaviour at different levels are necessary.

This study is not intended to integrate yet another more precise physical model of garment behaviour, but rather focus on the real-time constraints for the simulation and the visual cloth motion features to which an observer is sensitive. Most of the existing approaches use a general-purpose simulation method using collision detection and physical simulation for the whole garment. Unfortunately, simulations that simply calculate all potentially colliding vertices may generate a highly realistic movement, but do not provide a guaranteed frame time. A new simulation model should be implemented that avoids heavy calculation of the collision detection and particle system wherever possible.

Our assumption is that the whole cloth does not need to be simulated with a general-purpose simulation method; instead many optimizations can be made. For example, the trouser will never collide with the arms. Collision detection may be simplified by restricting the collision detection to only potentially colliding surfaces. Also, stretched garments do not need to be simulated with a complex physical method. It can be simply simulated by keeping an offset between the garment and the underlying skin surface. By making use of predetermined conditions between the cloth and the body model, the computation cost can be greatly reduced. We call these cloth simulation methods that are based on pre-determined conditions as "context-specific" simulators. In the remaining of this chapter, we will describe two of them.

The first approach [COR 02] works in a hybrid manner exploiting the merits of both the physically based and geometric deformations. Garments are segmented into pieces that are simulated by various algorithms, depending on how they are laid on the body surface and whether they stick or flow on it. Tests show that the method is well suited to fully-dressed virtual human models, achieving real-time performance compared to ordinary cloth-simulations.

The main idea of the second approach [COR 04] consists of developing a cloth simulator that can learn the cloth behaviour through a sequence of pre-computed cloth simulation. This approach enables us to simulate the garment features such as wrinkles, gathers… in real-time. We setup the problem as simulating the garments in two phases. The first phase, a rough mesh reproduces the dynamic behaviour of the garments. Its physical properties are defined by the pre-calculated sequence. In the second phase, the fine mesh simulates the details of the garments.

2 Previous Work

The history of research on real-time cloth is relatively recent. Researchers have concentrated mainly on two aspects of real-time cloth animation: simulating the physical properties of garments and collision handling.

2.1 Numerical solvers

Probably the most common technique for simulating the physical properties of clothes is the particle system. Simulation process is broken down into calculating the internal forces and solving the system of partial derivative equations (PDE). The latter point has attracted much interest in the field of real-time applications, since it requires high computation power.

The explicit Euler method [BAR 98] has been one of the first numerical solvers. Unfortunately, this method is notorious for its instability when using large time steps and stiff equations. Several improvements have been proposed to reduce instability, such as the Verlet integration [KAC 03] and the explicit Euler combined with inverse dynamics [PRO 95] [VAS 01]. Unfortunately, the simulation quality is sacrificed in favour of computation speed, due to the approximations employed in these models.

The implicit Euler method presented by Baraff et al. [BAR 98] performs the computation not by using the derivative at the current time, but the predicted derivative at the next time step. Unlike explicit Euler integration, the implicit Euler method offers higher stability while using large time-steps and clothes with stiff mechanical properties. A major drawback of this numerical solver, however, is the computation of a large linear system,

More recently, researchers worked on saving the computation time of the linear system solver. Desbrun et al [DES 99] proposed solving the linear system with a pre-computed inverse matrix. Kang et al. [KAN 02] proposed further optimization with a direct update formula for the positions and velocities of the cloth vertices. As indicated by the authors, these methods are not intended to provide a physically-correct cloth animation. Our approach to that problem is a data-driven mass-spring system: the simulation is corrected with a set of functions built from the pre-simulated animation. By doing so, we bring the deformation of the mass-spring system closer to the original cloth behaviour.

Another approach to fast garment deformations is hybrid approaches. Several researchers have noted that wrinkle deformation is geometric in nature and therefore can be computed with a geometric method. Wrinkles can be generated either by tessellating the cloth mesh [KAN 02] or rendering details on texture using bump mapping [HAD 99]. The main difficulty is defining a fold function that can simulate all kinds of wrinkle patterns. Moreover, determining the location and shape of wrinkles is left to CG artists. One of our contributions is a geometric wrinkling method that is “trained” by using a pre-simulated cloth sequence, rather than relying on users.

2.2 Collision handling

Collision detection is usually one of the bottlenecks in real-time animation. The problem is particularly acute in the case of clothes because these objects are highly deformable. Some methods exploit graphics hardware to compute collisions on bump maps [VAS 01]; others use implicit surfaces to check collisions on the body [RUD 00], or voxel trees, which partition the space hierarchically [MEY 00]. Using frame coherency to reduce computation cost has been explored by Zhang et al [ZHA 02]. In this work, we propose a data-driven collision detection method; we use the pre-simulated sequence to localize the collision checks to neighbouring cloth regions that have high probability to collide.

2.3 Data-driven approaches

The idea of building an interpolator from examples or pre-simulated data has proven to be a valuable tool in a variety of areas of CG, e.g. for modelling a variety of human body shapes and for motion synthesis. The basic idea is to build an interpolation space filled with a set of pairs of input parameters and the targeted graphical objects. During the run-time simulation, given an input, the interpolator will produce the corresponding output by interpolating the examples available in the interpolation space. Cloth animation depends on a high number of parameters and therefore a data-driven approach is difficult to adapt. Very recently, James et al. [JAM 03] resented such an approach, where physics-based deformation and collision detection are both handled in a unified framework. By blending of pre-computed orbits rather than using a mass-spring system, previous unseen results could be achieved, such as garments with stiff mechanical properties in real-time. However, they show little DoF to the clothes under simulation.

3 Hybrid approach using geometric and physics-based deformations

When observing a garment worn on a moving character, we notice that the movement of the garment can be classified into several categories depending on how the garment is laid on and whether it sticks to, or flows on, the body surface. For instance, a tight pair of trousers will mainly follow the movement of the legs, whilst a skirt will float around the legs. The first part of the study is to identify all the possible categories:

- Garment regions that stick to the body with a constant offset. In this case, the cloth follows exactly the movement of the underlying skin surface.
- Garment regions that flow around the body. The movement of the cloth does not follow exactly the movement of the body. In case of a long skirt, the left side of the skirt can collide with the right legs.
- Garment regions that move within a certain distance to the body surface are placed in another category. The best examples are shirtsleeves. The assumption in this case is that the cloth surface always collides with the same skin surface and its movement is mainly perpendicular to the body surface.

These three categories are animated with three different cloth layers (Figure 1). The idea behind the proposed method is to avoid the heavy calculation of physical deformation and of collision detection wherever possible, i.e. where collision detection is not necessary. The main interest of our approach is to pre-process the target cloth and body model so that they are efficiently computable during runtime. The skin and the garment are divided into a set of segments and the associated simulation method is defined for each. For each layer, we propose solutions and explain why they have been chosen.

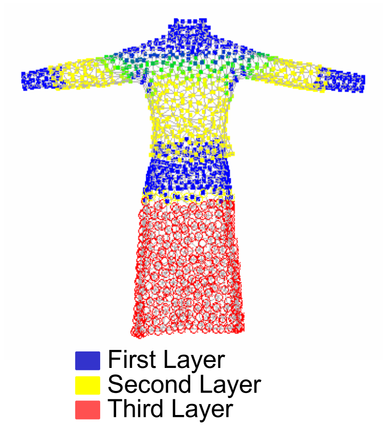


Figure 1: *Segmentation of garments.*

3.1 Layer 1: "Stretch clothes"

Tight clothes keep constant distances with the underlying skin surface. The deformation of this layer follows the deformation of the underlying skin. We choose to use the skin deformation method to animate this layer. This method does not involve any collision detection or physical deformation. It has no impact on the calculation time. Therefore, it is necessary to construct the skeletal information of this cloth layer. This information is defined by mapping the attachment information of the underlying skin to these cloth vertices. Each vertex of the garment mesh is associated to the closest triangle, edge or vertex of the skin mesh.

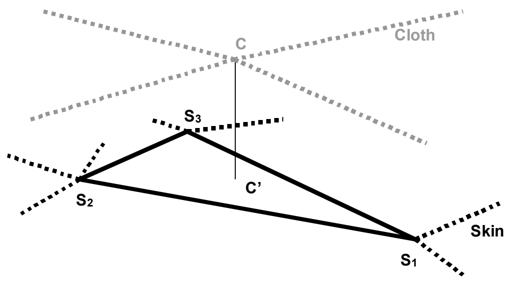


Figure 2: *Mapping of attachment information*

In the Figure 2, the garment vertex C is in collision with the skin triangle $S_1S_2S_3$. We define C' as the closest vertex to C located on the triangle $S_1S_2S_3$. We then define the barycentric coordinates of C' with S_1, S_2 and S_3. The attachment information of C is calculated by combining the attachment information of S_1, S_2 and S_3 weighted with the barycentric coordinates.

3.2 Layer 2 "Loose cloth"

For loose clothes, the relative movements of clothes to the skin remain relatively small, keeping a certain distance from the skin surface. To get an intuitive understanding of such cases, consider the movement of sleeve in relation with the arm: for a certain region of the garment, the collision area falls within a fixed region of the skin surface during simulation. With this in mind, the scope of the collision detection can be severely limited. A basic assumption made is that the movement of the garment largely depends on that of the underlying skin and yet it should not follow the skin surface rigidly. It is necessary to simulate the local displacement of the garment from the skin surface.

Two different methods have been developed, one for cloth deformation on the limbs (trousers and sleeves), the other one for the deformation of cloth on the trunk.

3.2.1 Sleeves and trousers

In this approach, we use the assumption that cylinders can approximate the limbs.

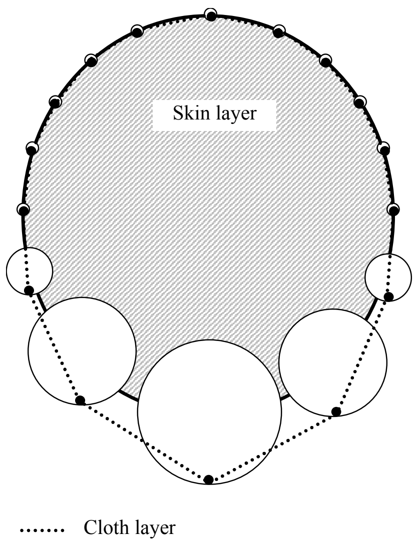


Figure 3: *Cross section of a limb with a garment*

Each vertex is kept on a disc that is attached to the skin.

- Initially, the position of the disc centres are calculated using the attachment information defined in the pre-processing stage. They are obtained by mapping the attachment information of the underlying skin to the cloth vertices. The axe of the discs is parallel to the limb joint.

- During simulation, the movement of the vertices on their disc follows the equation of the rigid body motion. Vertices move independently to each other. Using the 2nd Newton's law, velocity and position are easily calculated with gravity force. In case a vertex leaves its disc, a kinematic correction [WIT 01] is applied to the velocity and the position to put the vertex back on the disc surface. The Figure 3 shows a cross-section of a limb with a garment.

The size of the disc is a function of the angle between the cloth acceleration vector and the normal to the skin surface. The cloth acceleration vector is the subtraction of the gravity to the acceleration of the associated skin surface. This function is defined in a way that the size of the disc follows roughly the catenary shape, i.e. the shape of a hanging wire. It can be proven that if a heavy flexible cable is suspended between two points, then it takes the shape of a curve with the equation:

$$y = c + a\cosh\left(\frac{x}{a}\right).$$

Figure 3 gives an example of the variation of the size of the disc along the body surface.

We approximate the limbs (arms, legs…) to cylinders and then in order to determine the size of the discs for the cloth deformation we consider the equation of the circle (the cylinder that approximates the limb) and the equation of the catenary (the garment hold by the limb) as shown on the graph in Figure 4.

The equation of the half disc of diameter *2r* in Cartesian coordinate system (x, y) is:

$$y = \pm\sqrt{r^2 - x^2}\ .$$

The equation of the catenary in the same coordinate system (Figure 4):

$$y = a\left(\cosh\frac{x}{a} - \cosh\frac{r}{a}\right)$$

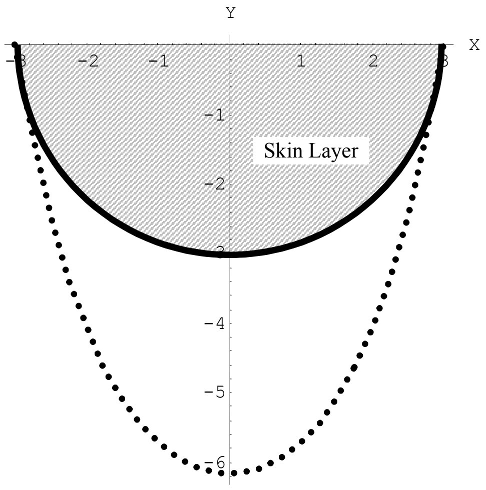


Figure 4: *Equation of the catenary and the disc*

where *2r* is the diameter of the limb and *a* is the scaling factor of the catenary curve. It is used to control the size (largeness) of the garment. We define the ideal size of the disc as the difference of the two equations above:

$$a\left(\cosh\frac{x}{a}-\cosh\frac{r}{a}\right)-\sqrt{r^2-x^2}$$

We define a quadratic equation to approximate the size of the disc. This equation should be fast to compute, as it will be used intensively to calculate the movement of every vertex at every frame. This approximation is done by fitting the polynomial function *P(x)* to the three points $P_0(x\text{=-}r)$, $P_1(x\text{=}0)$ and $P_2(x\text{=}r)$.

$$P(x)=C\left(\left(\frac{x}{r}\right)^2-1\right) \text{ where } C=a\left(1-\cosh\frac{r}{a}\right)+r$$

In this equation, *C* is a constant that is related to the size (largeness) of the cloth. This parameter is defined by the user at the pre-processing stage. The term $\left(\frac{x}{r}\right)^2$ is calculated with $\sin\theta=\frac{x}{r}$.

$\sin\theta$ is evaluated by projecting the acceleration component $\overline{OA}$ to the normal $\overline{ON}$ of the skin surface as shown in Figure 5. $\overline{OA}$ is the acceleration applied to the cloth. It is the subtraction of the acceleration of the skin vertex *O* to the gravity component.

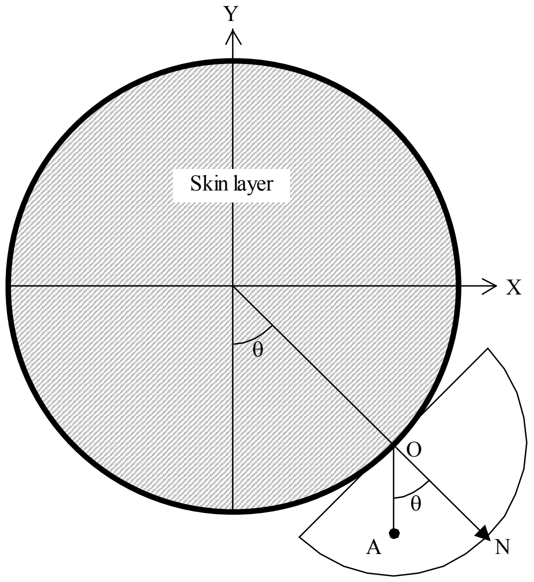


Figure 5: *Computation of the disc diameter*

This method ensures that the rest shape of the garment takes the shape of real garment hanged on a limb. All the discs containing the vertices are parallel to their corresponding limb joints. This ensures that every vertex of the same limb will fall down to the same side on the limb, even in case the direction of the limb is almost vertical. If the limb is perfectly vertical, the result is unpredictable. However, this case never happens in practice.

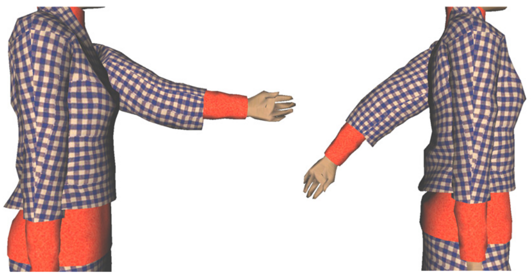


Figure 6: *Deformation of sleeves*

In this method, no collision detection on surfaces is necessary. Our particle system is very simple, each vertex moves independently. Therefore, the acceleration applied on vertices is constant. The stability of the system does not depend on the time step duration. The resulting motion of the garments provides the appearance of dynamic movements as well as the shape of real clothes. An example of a sleeve deformation is shown on Figure 6.

3.2.2 Clothes on the trunk

The hybrid technique described in Section 3.2.1 works well if the skin surface can be approximated by a moderately thin cylinder and the movement of the garment is more or less perpendicular to the skin surface. This assumption is true for the arms and legs; the friction prevents the garments to move along the limbs. In such cases, vertices are considered to move independently of each other. However, such an assumption cannot be made for garment regions around the trunk where, unlike the arm, the movements of the garment are driven not only by the torso but also by the potentially complex movement of the shoulder and the arm. To get an intuitive understanding of the problem, consider a shirt worn by a virtual human who is raising its arms. The whole garment around the torso part moves along the vertical direction as the arms are raised. This implies the necessity to simulate the elastic property of the tissue. Therefore, a dedicated method for cloth deformation on the trunk has been developed.

A simplified mesh composed of 42 nodes is animated using a simplified mass-spring system. The movement of these nodes is driven by two interactions: gravity and springs in connection with neighbouring nodes. Each of these nodes is a control point used to deform the cloth mesh on the trunk. These control points are uniformly placed on the trunk, they form a grid of 3x4x3 points. Six additional control points are attached to the shoulder. These control points maintain the grid mesh on the trunk and prevent the mesh from falling due to the gravity. Each control point is included in a half-sphere that is attached to the skeleton. The collision detection algorithm verifies if each control point is contained within its corresponding hemisphere. In case the control point is not in its hemisphere, a kinematical correction is applied on the position and the speed.

The cloth mesh is deformed with the Freeform Deformation method [BAR 84] using the position of the nodes of the simplified mesh. Our implementation uses a 3x4x3 control point lattice for the FFD and uses Bernstein polynomials as the basis functions. The example in shows the grid, composed of the 36 green nodes. The weight values associated to the vertices, deformed by the FFD, are pre-calculated.

3.4 Layer 3 “Floating cloth”

Layer 3 is composed of vertices that freely float around the body. This will take care of cases, such as a large skirt floating around the legs. Any part on this skirt can collide with any part of the leg. The simulation of this layer uses a classical approach with particle system and collision avoidance.

- **Particle system**: We use a simple mass-spring system. The simulation is performed using the Implicit Euler Integration proposed by Barraf et al. [BAR 98] and Volino et al. [VOL 00]. The garment is modelled by a simple mass-spring system. We consider two interactions: gravity and forces applied by the springs joining the particles.
- **Collision response**: Calculating collision detection between the cloth and the skin mesh would not be feasible in real-time, therefore we use a simplified model of the body. Given the assumption that the floating clothes are mainly skirts, the collision detection is calculated for the legs only and two cylinders model each leg. By simply calculating the distances between the skirt and the four leg segments, the collision detection can be performed. It is possible to further optimize by restricting the number of collision distances that we compute for each segment. During the pre-processing stage, a list of possible colliding vertices is defined for each segment. This list is defined by calculating the distance between the vertices and the legs and the normal orientation for each vertex on the skirt. The collision response consists of applying a kinematical correction to the position and velocity of the colliding vertices.

3.5 Computation speed

Our method has been tested on several dressed virtual humans. Figure 7 illustrates how computational time changes with an increasing number of cloth triangles. The code was executed on a 1GHz PC with 512 MB RAM and a GeForce2 graphics card. The computational times do not include the deformation of the avatar skin and the real-time rendering. A simulation step corresponds to 0.04 seconds worth of real-time animation.

The deformation of Layer 3 is in average thirteen times slower than the deformation of the Layer 2. This is because Layer 2 uses an optimized model with a simplified collision detection method and no integration method. The deformation of Layer 1 is even faster because it uses skeletal deformation. The Layer 3 algorithm is able to simulate a maximum of 1,000 non-tessellated polygons in real-time. In most cases, this is sufficient to produce aesthetically pleasing results. Another advantage is that most of the clothes can be animated using Layer 2. Layer 3 is used in only a limited number of cases, such as large skirts or large trousers.

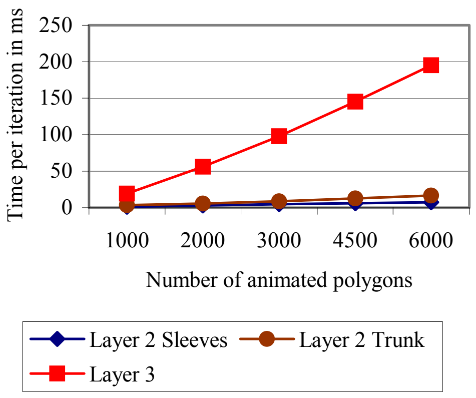


Figure 7: *Computing time per iteration versus number of triangles.*

4 Data-driven cloth simulator

We present a data-driven method for simulating clothes worn by 3D characters in real-time [COR 04]. To effectively optimize the physics-based deformation, which is the bottleneck of the simulation, we use a coarse representation of the cloth mesh to drive the gross behaviour in simulation. We consider that the gross cloth behaviour is driven mainly by two separable contributions: the skeleton-driven movement of the character and the mechanical properties of the cloth. This consideration was partly inspired by the hybrid real-time simulation method proposed in the previous section [COR 02], where a hybrid deformation method is used to combine dynamic surfaces with skeleton-driven deformation (SDD). Unlike that method, however, our method exhibits significantly more efficient and realistic behaviour. This effect is achieved by focusing on the analysis of cloth movements in relation to its associated skin surface, and adopting a learning strategy. The idea is to use the analysis of the pre-simulated sequence to identify the region largely explained by joint movement and to replace the physics based simulation with geometric methods wherever possible.

In our approach, the key ingredients of the new technique are associated with different facets of cloth simulation: First, our novel collision detection prunes out unnecessary collision tests by tightly localizing potentially colliding regions through the analysis of the cloth movement in relation to the skeleton. Second, we use the pre-simulated sequence to approximate the dynamic behaviour of the coarse mesh geometrically wherever possible. Finally, fine details such as wrinkles are also simulated in a data-driven manner, by using the pre-simulated cloth sequence as examples. Subsequently, real-time animation of fully dressed human could be generated, which would be suitable for applications such as games where visual plausibility is more important than accuracy.

4.1 Simulation of the gross behaviour

Due to the computational expenses of solving the full numerical system of the physics-based deformation, we seek simplifications by constructing a coarse mesh representation of the garment. The coarse mesh is used to deduce the gross behaviour of the cloth in a data-driven manner, based on the input pre-simulated sequence. A number of optimization strategies are adopted: The two following sections describe a pre-processing that constructs and segments a coarse mesh representation into different region types. We then describe in the next two sections the spring-mass system and collision handling of the coarse mesh at each frame of the simulation. Also described is the runtime process.

4.1.1 Construction of the coarse mesh

We begin by constructing a coarse representation of the given cloth model that will drive the gross behavior of the simulated garment.

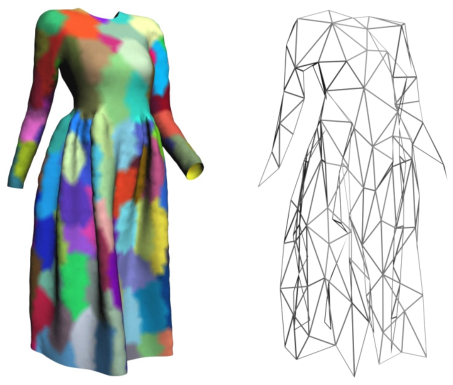


Figure 8: *The fine and the coarse mesh.*

It consists of two following steps: (1) The cloth surface is partitioned into a set of patches as shown in Figure 8. (2) A coarse mesh representation is obtained by combining a set of vertices in a patch into a single mass point located at the center.

4.1.2 Identifying cloth-to-joint relations and region types

Next we carry out cloth-to-skin (or body) attachment through skin fitting, by which the skinning data on the cloth mesh are approximated in such a way that the skinning-driven cloth shape best fits the simulated cloth shape throughout the whole pre-simulated sequence. The basic idea is to use the pre-simulated results as examples and find the error-minimizing skin data through optimization (Figure 9). An optimization approach, such as the one presented by Mohr et al. [MOH 03], has been adopted here.

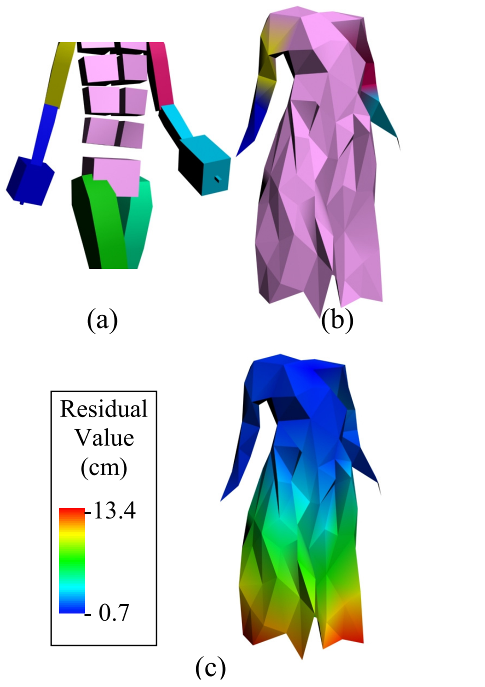


Figure 9: *(a) and (b) influence of the joints on the dress shown in colour, (c) quality of the fitting of the SDD data.*

The residual values of the fitting provide useful information on how the garments behave in relation to the body. In our approach, three regions are identified from the residual values of the skin fitting process (Figure 10): those that potentially interact with several joints, those that are loosely attached to the skeleton and those that are rigidly attached to the skeleton.

The deformation of tight regions is directly computed with the Skeleton Driven Deformation (SDD). The use of SDD for these regions makes it possible to reduce the number of mass points in the mass-spring system.

High residual values indicate much less dependency on a specific body region of the cloth movement. Therefore, an additional collision check is required to handle the interaction of the clothes with the whole body skeleton. A list of potentially colliding body patches is defined by selecting those that approach within a certain distance of the floating regions during the pre-simulated cloth sequence.

The three garment regions

🟦 Garments rigidly attached to the skeleton through SDD

🟩 Garments for which the collision detection is made with collision hulls only

🟥 Garments for which the collision detection is made with collision hulls and a predefined list of potentially colliding joints

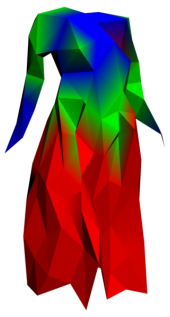


Figure 10: *The three regions computed on an analysis of the residual values..*

4.1.3 Data-driven post-correction of the coarse mesh

At each frame of the simulation, we compute the coarse mesh by a mass-spring system with the implicit Euler numerical solver [BAR 98]. The simulation run on the coarse mesh hardly reproduces the gross movement of the original cloth because the initial mesh has been significantly simplified (from 4000 to a few dozen vertices) and the topology has been modified. Moreover, unlike the simulator used for the pre-simulated cloth sequence, the simplified mass-spring model does not accurately simulate the bending and shearing properties of the fabrics [VOL 00].

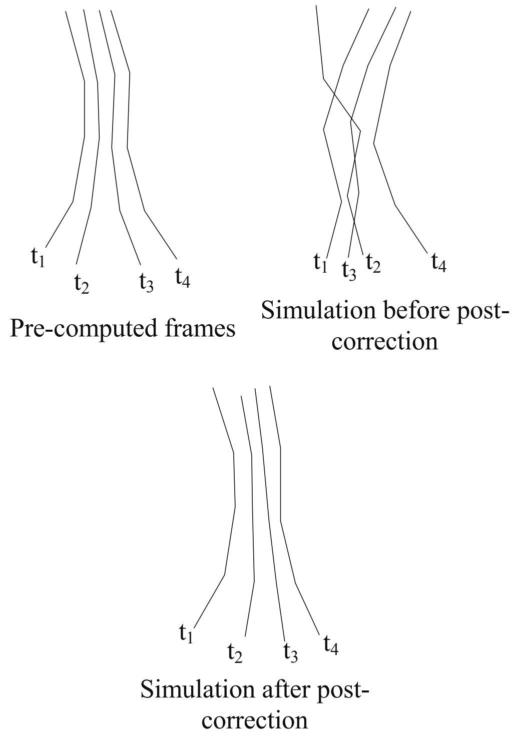

t_1
t_2
t_3
t_4
Pre-computed frames
Simulation before post-correction
Simulation after post-correction

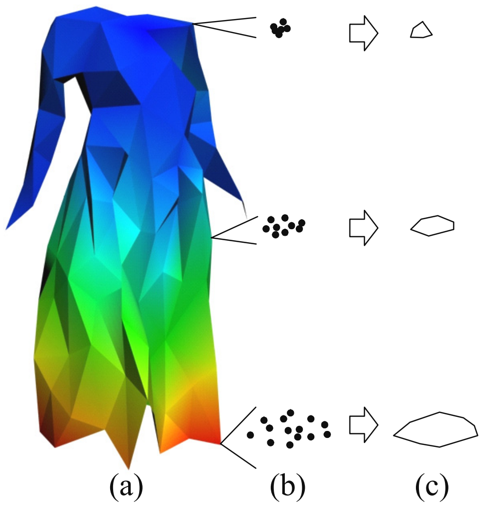
Figure 11: *Post-correction of the mass-spring system.*

We approach the problem by modifying the behavior of the mass-spring system through a fix-up process (similar to [MEY 00]) where the position and velocity of the coarse mesh vertices are modified in order to maintain the cloth shape as close as possible to the original one (Figure 11).

Ideally, the local shape (e.g. position of the vertices in relation to their neighbors) should be a blend of those of the pre-simulated animation. This is achieved by constructing a set of functions of local shape deformation. Post-correction is accomplished with a function that evaluates the "ideal" position of the vertex given the position of its neighbors connected by the edges. The construction of these interpolators is described in detail in [COR 04]

4.1.4 Collision hulls

To prune unnecessary collision tests, we pre-compute what we term "collision hulls" that exploit the skin-to-cloth relation obtained from the pre-simulated sequence. These are built once at the beginning of the simulation (prior to the runtime simulation) after the SDD has been computed on the coarse mesh, using the pre-simulated sequence. At each pre-simulated frame, we calculate the difference between the SDD motion model and the pre-simulated cloth model in the local coordinate system of the SDD. After a sweep, we get a set of points that cover the path a patch takes during the simulation. The smallest convex hull (Figure 12 (b) and (c)) that contains all these points is generated for every patch using the "Quickhull" algorithm presented by Barber et al [BAR 96].



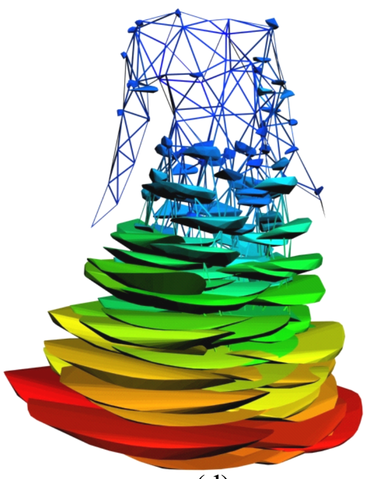


Figure 12: *Post-correction of the mass-spring system.*

Collision handling at runtime consists of correcting the position of coarse mesh vertices after every simulation step so that they remain inside their respective hulls. Thus, collision detection returns to computing the inclusion of the particle in its associated hull; the Gilbert-Johnson-Keerthi algorithm [GIL 88] is ideally suited to this task. We used constrained dynamics [WIT 01] to handle the collision response (i.e. modification of position and velocity in response to collision detection). Collision detection is also computed between floating regions and skeleton joints.

4.1.5 Runtime computations

The real-time computation of global cloth movements is obtained with a mass-spring system together with the collision response and post-correction described above. The runtime computation of the coarse mesh is obtained in the following order:

- Mass-spring computation
- Post-correction
- Collision response on hulls
- Collision response on floating garments

4.2 Generating garment details

So far we have shown the first part of our simulation, that is, the coarse level simulation. We continue now to describe the second part of the simulation, by which detailed cloth shape such as wrinkles or folds are depicted. Again, the main challenge here is obtaining the highest possible realism while maintaining acceptable computation load, in order to meet the real-time requirements.

As recognized in earlier works [HAD 99] [KAN 01], wrinkles can be efficiently animated with a geometric method as they are geometric in nature. Unlike previous methods, however, our wrinkling function is not hand-drawn, nor geometrically approximated, but rather trained from on the analysis of the pre-simulated sequence.

In this work, we choose to represent the wrinkle displacement in the local coordinate system used for SDD. This makes our wrinkle parameterization invariant of all joints of higher hierarchy than the currently influencing joint.

Several techniques exist for shape interpolation using examples, such as RBF or parametric interpolation. We have used linear interpolation in which coefficients are defined by multi-linear regression on the pre-simulated animation, since it provides satisfactory results at a very low computation cost. For every vertex *x* in a patch, the interpolator function takes the associated mass point in the coarse mesh, and its neighbors as input. To calculate the position of *x* from the input, the wrinkle interpolator interpolates the positions of the coarse mesh points, weighted by coefficients determined the regression model on the pre-computed animation (Figure 13). For further details on the construction of the wrinkle interpolator, we refer to [COR 04].

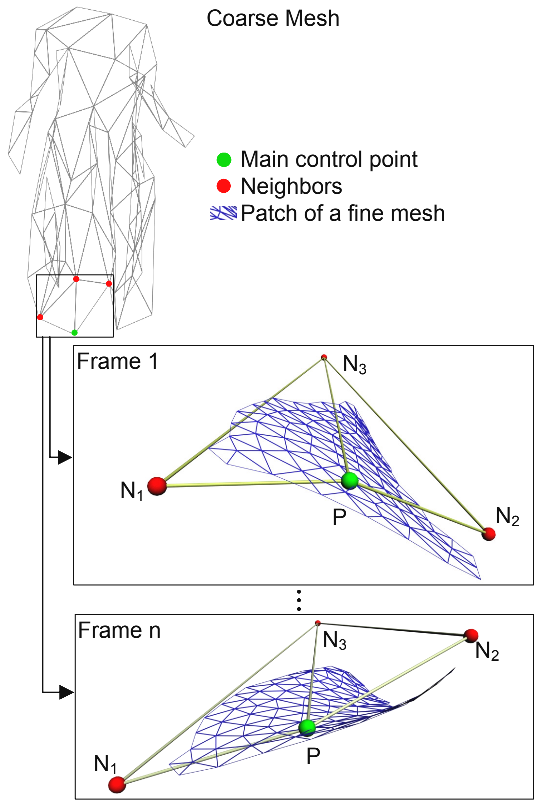


Figure 13: *Shape of the patch with respect to the positions of the control point P and its neighbours N1, N2, and N3.*

4.3 Results

Our simulator behaves fairly well on a wide variety of clothes, including those with highly stiff mechanical properties. Figure 14 show the pre-processing and run-time simulation results for the "cocktail" dress the "jean" outfit. Moreover, performance will increase due to the fact that the smallest number of triangles will be processed for the real-time rendering.

However, the method may introduce flaws in simulation for some tight clothes, due to the approximate handling of collision detection. For some body movements, the skin surface may slightly intersect the cloth surface. Similarly, the same problem may arise for self-collisions on clothes. The deletion of the skin triangles covered by the garment surface can partially correct this drawback.

Another drawback of the simulator is that the quality of the simulation depends on the number and variety of examples – the pre-simulated sequence in our case. We refer to [COR 04] for a detailed description of the simulator performance when using different pre-computed cloth simulation sequences.

5 Conclusion

Both techniques rely on the classification of the cloth vertices into the tight, loose and floating regions (see Section 3 and Section 4). Such classification has permitted to utilize methods that are optimized for specific cloth behaviors (i.e. context-specific simulators). These specialized methods are faster than general, versatile ones because they are based on strong assumptions about the movements of the garments in relation to the skin surface and thus they avoid doing unnecessary computation.

Both techniques share the same idea: the cloth surface is attached to the skeleton and its deformation is computed in relation to it. In other words, garments are considered as a second skin layer and their movements closely related to the skeleton animation. With this idea in mind, the mechanical model and the collision detection have been adapted.

However, the cloth simulation is restricted to clothes worn on bodies. While offering high computation speed, the cloth simulator cannot handle some cloth movements such as those appearing during dressing or undressing. More generally, the clothes are unable to interact with objects other than those that have been taken into consideration during the pre-processing phase. The list of objects that can potentially interact with clothes and the way these objects interact are defined at the pre-processing stage and cannot be changed during the real-time simulation.

Acknowledgements

This work is supported by the Swiss National Science Foundation (SNSF). Thanks to Marlène Arévalo and Christiane Luible for their design work to the garment models illustrating this chapter.

References

[ALE 02] Alexa, M., "Linear Combination of Transformations", SIGGRAPH 2002 Conference Proceedings, Annual Conference Series, ACM Press, Vol. 21(3), pp. 380-387, 2002.

[BAR 98] Baraff, D., and Witkin, A., "Large steps in cloth simulation", ACM Transactions on Graphics, Proceedings of ACM SIGGRAPH, ACM Press, pp. 43-54, 1998

[BAR 96] Barber, C. B., Dobkin, D.P., and Huhdanpaa, H.T., "The Quickhull Algorithm for Convex Hulls", ACM Transactions on Mathematical Software, ACM Press, Vol. 22(4), pp. 469-483, 1996

[BUR 93] R. L. Burden, J. D. Faires, "Numerical Analysis, Fifth Edition", published by PWS Publishing, ISBN 0-534-93219-3, 1993.

[COR 02] Cordier, F., Magnenat-Thalmann, N., "Real-time Animation of Dressed Virtual Humans", Eurographics, Blackwell publishers, pp 327 – 336, 2002.

[COR 04] Cordier F., Magnenat-Thalmann N., "A Data-driven Approach for Real-Time Clothes Simulation", accepted to the 12th Pacific Conference on Computer Graphics and Applications (Pacific Graphics 2004), Cohen-Or, D., Ko, H.-S., Terzopoulos, D. and Warren, J., eds., October 6-8, 2004, Seoul Korea.

[CHO 02] K.-J. Choi, H.-S. Ko, "Stable but Responsive Cloth", ACM Transactions on Graphics, Proceedings of ACM SIGGRAPH 2002, ACM Press, 21, pp. 165-172, 2002.

[DES 99] Desbrun, M., Schröder, P., and Barr, A. H. "Interactive Animation of Structured Deformable Objects", In Graphics Interface'99 proceedings, Morgan Kaufmann, pp. 1-8, 1999.

[GIL 88] Gilbert, E. G. , Johnson, D.W., and Keerthi, S. S., "A fast procedure for computing the distance between complex objects in three-dimensional space", IEEE Journal of Robotics and Automation, published by IEEE Press, 4(2), pp. 193-203, 1988

[HAD 99] Hadap, S., Bangarter, E., Volino, P., Magnenat-Thalmann, N., "Animating Wrinkles on Clothes", IEEE Visualization '99. San Francisco, USA, IEEE Press, pp. 175-182, 1999.

[JAM 03] James D. L., and Fatahalian, K., "Precomputing Interactive Dynamic Deformable Scenes", ACM Transactions on Graphics, ACM Press, Vol. 22(3), pp. 165-172, 2003.

[KAC 03] Kacic-Alesic, Z., Nordenstam, M., Bullock, D., "A practical dynamics system", ACM SIGGRAPH/Eurographics Symposium on

Computer Animation, ACM Press, pp. 7—16, Jul 2003.

[KAN 02] Kang, Y.-M., Cho, H.-G., "Bilayered Approximate Integration for Rapid and Plausible Animation of Virtual Cloth with Realistic Wrinkles", Computer Animation 2002, Switzerland, IEEE Press, pp. 203, 2002.

[KAN 01] Kang Y.-M., Choi J.-H., Cho H.-G., Lee D.-H., "An efficient animation of wrinkled cloth with approximate implicit integration", The Visual Computer Journal, Spinger-Verlag, 2001.

[MEY 00] Meyer, M., Debunne, G., Desbrun, M., Barr, A. H., "Interactive Animation of Cloth-like Objects in Virtual Reality". Journal of Visualization and Computer Animation, John Wiley & Sons, 2000.

[MOH 03] Mohr, A., and Gleicher, M., "Building Efficient, Accurate Character Skins from Examples", ACM Transactions on Graphics, ACM Press, Vol. 22(3), pp. 165-172, 2003

[FLA 88] Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T., “Numerical Recipes in C, The art of scientific computing”, Cambridge University Press, 1988.

[PRO 95] Provot, X., "Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior", Graphics Interface'95 proceedings, A K Peters, pp 147-154, 1995.

[RUD 00] Rudomin, I., Meln, M., "Multi-Layer Garments Using Hybrid Models", Visual 2000, Springer, pp. 118, 2000.

[TER 88] D. Terzopoulos, k. Fleischer, "Deformable Models", The Visual Computer, Springler-Verlag, Vol.4 (6), pp.306-331, 1988.

[VAS 01] Vassilev, T., Spanlang, B., "Fast Cloth Animation on Walking Avatars", Eurographics, Blackwell Publishers, 2001.

[VOL 00] Volino, P., Magnenat-Thalmann, N., "Virtual Clothing - Theory and practice", Springer Verlag, ISBN: 3-54067-600-7, Dec 2000.

[WIT 01] Witkin A., "Constrained Dynamics", Physically-Based Modeling SIGGRAPH 2001 Course Notes.

[ZHA 02] Zhang, D., Yuen, M., "A Coherence-based Collision Detection Method for Dressed Human Simulation", Computer Graphics Forum, Blackwell publishers, Vol. 21(1), pp. 33-42, 2002.

[BAR 84] A. H. Barr. Global and local deformations of solid primitives. SIGGRAPH 84 Conference Proceedings, Annual Conference Series, pages 21–31. ACM SIGGRAPH, Addison Wesley, July 1984.

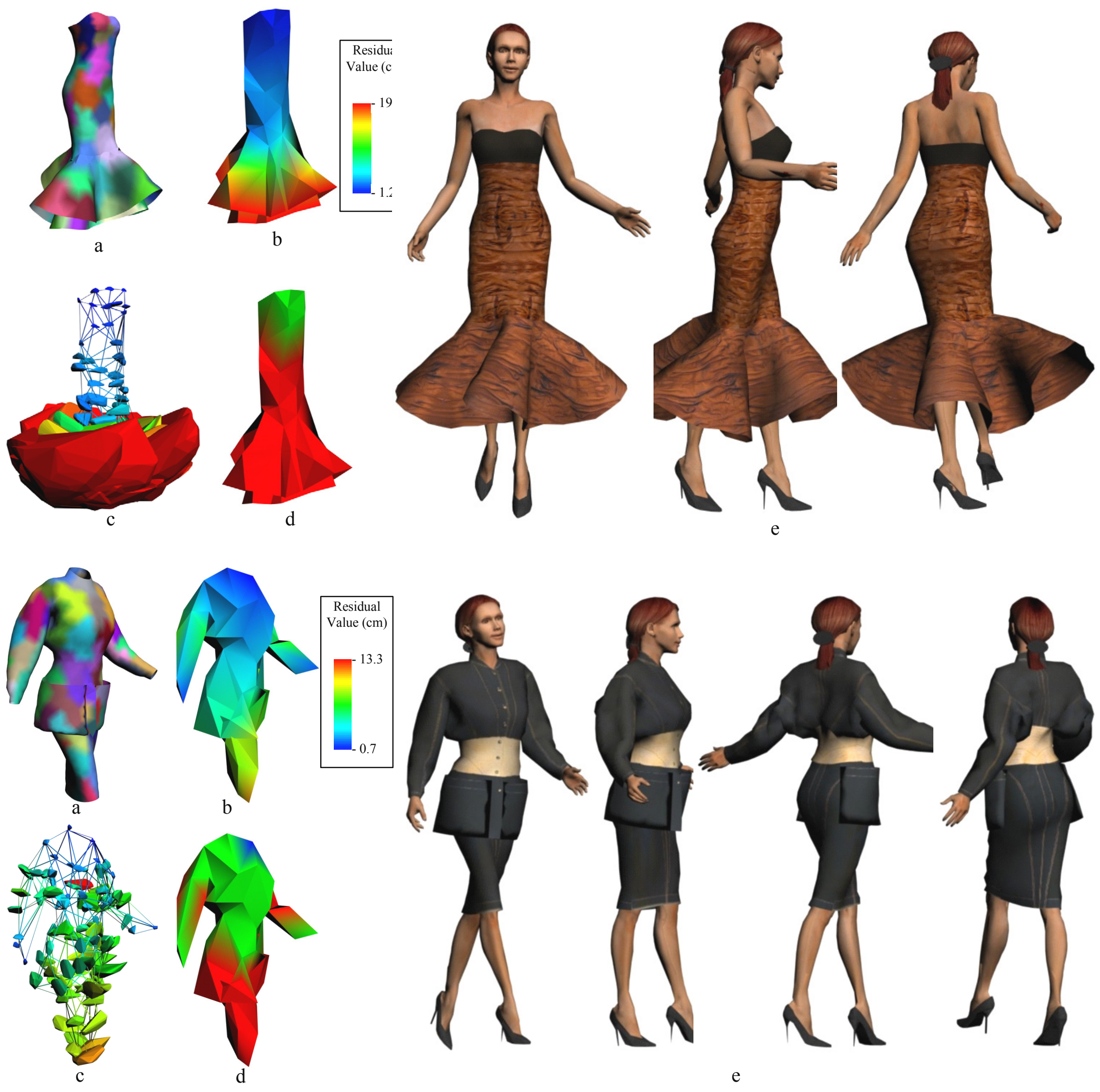


Figure 14: *Simulation of cocktail dress and jean outfit: segmentation (a), analysis of vertex movements (b), identification of the three regions (c), the resulting coarse mesh with collision hulls (d), samples of real-time animation (e).*

Case study: the Pompeii Project

(Tutorial Notes)

Nadia Magnenat-Thalmann, George Papagiannakis, Frederic Cordier

MIRALab, CUI, University of Geneva, Switzerland

E-mail: {thalmann, papagiannakis, cordier}@miralab.unige.ch

Abstract

In this chapter we present our work on the LIFEPLUS EU IST project. LIFEPLUS proposes an innovative 3D reconstruction of ancient frescos-paintings through the real-time revival of their fauna and flora, featuring groups of virtual animated characters with artificial life dramaturgical behaviours, in an immersive AR environment. The goal of this project is to push the limits of current Augmented Reality (AR) technologies, exploring the processes of narrative design of fictional spaces where users can experience a high degree of realistic interactive immersion. Based on a captured/real-time video of a real scene, the project is oriented in enhancing these scenes by allowing the possibility to render realistic 3D simulations of virtual characters in real-time.

1 Introduction

Since antiquity, images were used as records of both events-lifestyles, as well as decorations. The possibility of reviving them, adds a new dimension in understanding our past. However, the recreation of historic environments for serious study, education and entertainment is not new, Arnold [ARN 00], although the methods for achieving the objectives have evolved considerably over time. Before the days of widespread books and printing, story tellers would conjure up visions of events and places, providing their listeners with an impression of realities (often augmented realities) elsewhere in time and space. Theatre, fine art and cinema have added to the richness of the explicit visual experience available to the viewer. They have made the interpretations of history more accessible to the general public, but at the same time narrowing the individual's scope for personalized, interactive experience and visualization of the description of it. Historical frescos are a unique arrangement of "mise-en-scene" elements that enhance the user experience by creating a set of compelling narrative patterns, alas however in a static, two-dimensional way. Mixed Realities, Milgram and Kishino [MIL 94], and their concept of cyber-real space interplay invoke such interactive digital narratives that promote new patterns of understanding in various contexts. In the context of cultural heritage sites such as the ancient city of Pompeii, would like to observe and understand the behaviors and social patterns of living people from ancient Roman times, superimposed in the natural environment of the city. In industrial environments, we would like to 'employ' virtual workers for training and maintenance reasons, in order to assist real ones. Since recently, AR Systems had various difficulties to manage such simulations in a fully interactive manner, due to hardware & software complexities in AR enabling technologies, Azuma et al [AZU 01].

2 Related Work

A number of projects are currently based on AR integrated platforms, exploring a variety of applications in different domains such as medical, ART [ART 04], cultural heritage, Stricker et al [STI 01] and Papagiannakis et al [PAP 02], training and maintenance, Schwald et al [SCH 01] and Wohlgemuth and Triebfürst [WHO 00], and games, Thomas et al [THO 00]. Special focus has recently been applied to system design and architecture in order to provide the various AR enabling technologies a framework, Gamma et al [GAM 94], for proper collaboration and interplay. Azuma, [AZU 01], describes an extensive bibliography on current state-of-the-art AR systems & frameworks. However, few of these systems take the modern approach that a realistic mixed reality application, rich in AR virtual character experiences, should be based on a complete VR Framework (featuring game-engine like components) with the addition of the "AR enabling Technologies" like a) Real-time Camera Tracking b) AR Displays and interfaces c) Registration and Calibration.

3 A/R Framework Components

Our AR platform is based on the VHD++, Ponder et al [PON 03], component-based framework engine developed by VRLAB-EPFL and MIRALab-UNIGE which allows quick prototyping of VR-AR applications featuring integrated real-time virtual character simulation technologies. The framework has borrowed extensive know-how from previous platforms such as presented by Sannier et al [SAN 99]. The key innovation is focused in the area of component-based framework that allows the plug-and-play of different heterogeneous technologies such as: Real-time character rendering in AR, real-time camera tracking, facial simulation and speech, body animation with skinning, 3D sound, cloth simulation and behavioural scripting of actions. To meet the hardware requirements of this aim, a single DELL P4 M50 Mobile Workstation was used, with a Quadro 4 500 GL NVIDIA graphics card, a firewire Unibrain Camera or USB Logitech web camera for fast image acquisition in a video-see-through TekGear monoscopic HMD setup, for advanced immersive simulation. Our previous efforts were based on a client-server distributed model, based on 2 mobile workstations. To achieve the requirement of 'true mobility', a single mobile workstation is used in our current demonstrations, after improvements in the streaming image capturing and introduction of hyper-threading in the platform code.

4 Real-time Markerless Camera Tracking

Real-time markerless camera tracking presents two main problems, namely the absence of easily recognisable markers and a demand for high-speed computation. Using markerless tracking is often a necessity as applying markers within the tracking area is not suitable or possible (especially on cultural heritage sites). This then requires accurate tracking to be done with only natural features present, often with sharply varying light sources, shadows, motion blur and occlusions. Performing this tracking in real-time necessitates the use of algorithms specially adapted to fast operation, and thus disqualifies many algorithms that are perfectly suitable in offline applications.

The LIFEPLUS application utilizes the 2D3 camera tracking solution, 2d3 [2D3 04], based on the approach that the system should be able to self-initialize anywhere within the tracking environment without any intervention from the user. In effect this means that instead of calculating relative changes in rotation and translation, we calculate absolute rotation and translation for every frame. This has the advantage of avoiding the problem of drift, and also ensures instant recovery after tracking was lost due to excessive motion blur or occlusion.

5 Integrated Research in Cloth Simulation

There is still a gap between the time expensive techniques (Figure 1) that bring simulation accuracy and duplication of actual fabric mechanical parameters, and efficient techniques that are able to manage complex animated garments with simplified mechanical models. In order to define realistic cloth simulation systems that are able to simulate complex garments realistically while keeping a reasonable computation time, a deeper study of the cloth model and the identification of its movement behaviour at different level are necessary [12]. This study should not intend to integrate yet more precisely the parameters measured for given fabric materials, but rather focus on the real-time constraints for the simulation and the visual cloth motion features to which the observer is sensitive. A new simulation model has been implemented that avoids heavy calculation of collision detection and particle system wherever it is possible.

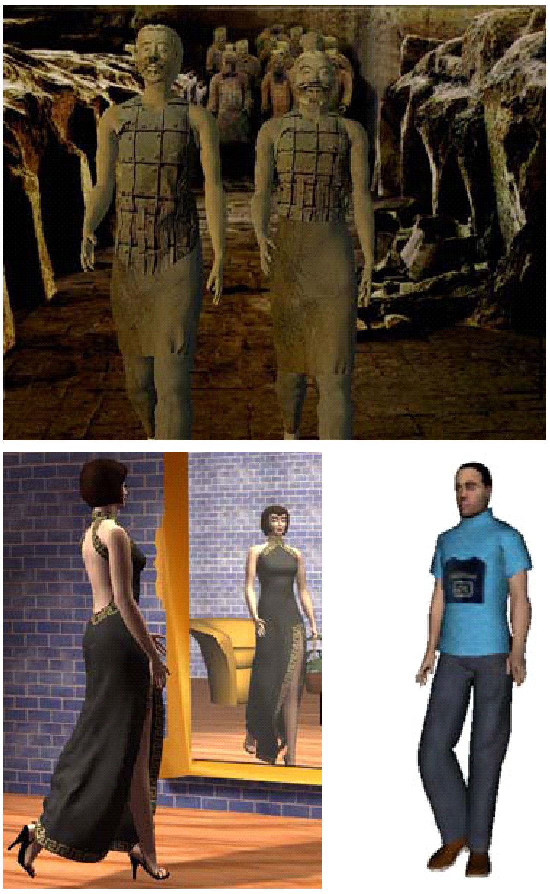


Figure 1: *Virtual Human Clothing.*

One of the most challenging research areas in this context is in developing a robust methodology on simulating clothes in real-time performance. We have developed a novel approach exploiting the merits of geometric deformations and predetermined conditions between the cloth and the body model. When observing a garment worn on a moving character, we noticed that the movement of the garment could be classified into several categories depending on how the garment is laid on the body surface and whether it sticks or flows on it. For instance, a tightly worn trouser will mainly follow the movement of the legs while a skirt will flow around the legs. The first part of the study was to identify all the possible categories:

a) Clothes that stick to the body with a constant offset. In this case, the cloth will follow exactly the movement of the underlying skin surface.

b) Clothes that flow around the body: the movement of the cloth does not follow exactly the movement of the body. In case of a long skirt, the left side of the skirt can collide the right legs.

c) Clothes that move within a certain distance to the body surface will be put into another category. The best example is the sleeves of a shirt. The assumption in this case is that the cloth surface always collides the same skin surface and its movement is mainly perpendicular to the body surface.

The idea behind the proposed method is to avoid heavy calculation of collision detection wherever it is not necessary. Using a versatile physical-based method for the whole garment implies huge calculation because of the dimension of the particle system and the number of polygons for collision detection. The main interest of our approach is to pre-process the target cloth model and segment the cloth in order to define the parts where we trade the simulation quality for real-time performance. Simulations that simply calculate all potentially colliding vertices may generate a highly realistic movement, with no guaranteed frame time. Following is a description of the methods that are employed for real-time clothing:

Geometric deformation: A simple geometric deformation is employed for clothes that stick to the body as they are deformed the same way as the underlying skin. No collision detection is required; the method will keep a constant offset between the cloth and the underlying skin.

Hybrid deformation: For the clothes that sweep on the body, a hybrid deformation is applied considering the fact that the movement will mainly be sweeping, and not complex movements such as draping, buckling, or wrinkle resilience. With the assumption of its perpendicular movement to the skin surface, each cloth vertex is modelled as a particle freely moving inside a sphere following the equation of rigid body motion. The spheres are attached to the skin and follow its movements rigidly. In case the particle leaves the spheres, a kinematic correction is applied on the position and the velocity.

6 Case Study: Early Mixed Realities simulations in ancient Pompeii

Our early results (the project started 1st March 2002) span across the frame of non-real-time mixed realities simulations to real-time VR interactive experiences. In greater detail, we have tested our methodologies-plug-ins, SDKs and platform in the following 3 example scene simulations featuring behavioured virtual Pompeian characters. In Figure 2, the screenshots depict part of our test rendered sequences, where virtual clothes, hair, speech, body and facial animation are tested using 3D Studio Max [MAX 04] and MIRALab dedicated virtual human plug-ins.

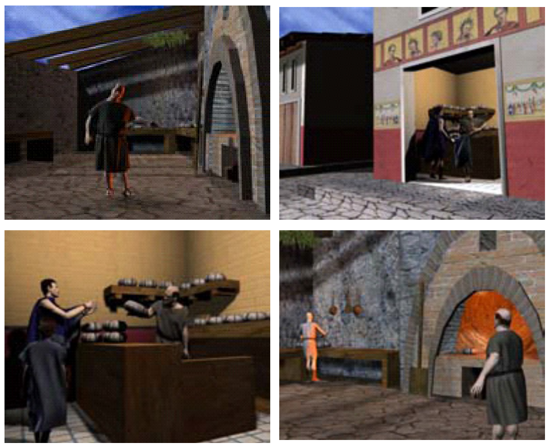


Figure 2: *At the Pompeian bakery in non-real-time VR*

In Figure 3, screenshots of the real-time simulation based on our framework platform VHD++ are shown, where real-time scene fly-through, virtual body and facial animation are illustrated.

In Figure 4, the 2D3 [2D3 04] Boujou^{TM} software was used to perform offline camera tracking and then the mixed reality simulations of virtual life were performed. Thus the virtual humans were correctly 'registered' with the 'real scenes' from the various 'real' video sequences using 3DSMax and the dedicated MIRALab plug-ins. The screenshots are excerpts from the complete simulation sequences.

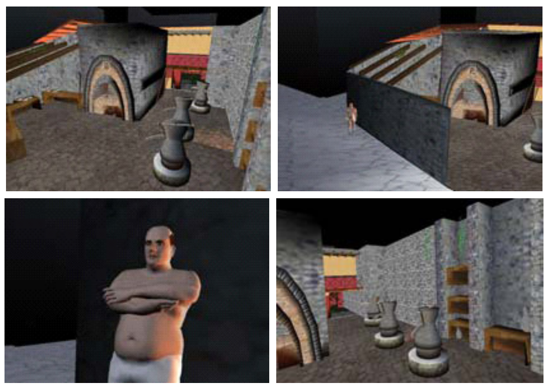


Figure 3: *At the Pompeian Bakery in Real-time VR*

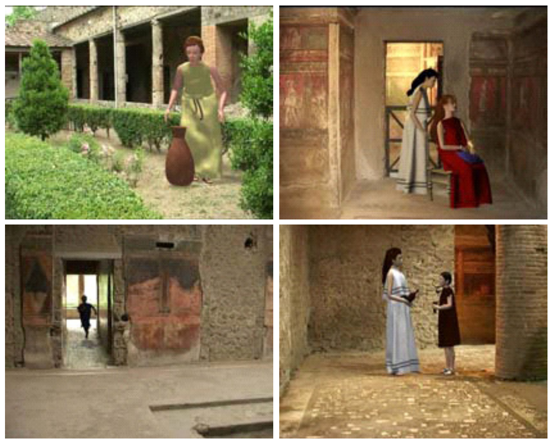


Figure 4: *Mixed Reality offline simulations in various Pompeian residences*

7 Heritage AR Simulation: Pompeii and the thermopolium of Vetutius Placidus

In order to validate that our integrated AR framework for virtual character simulation operates in different environments, we have tested the system directly in the ruins of Pompeii. However, in order to further continue AR tests in our, a real 'maquette' was constructed in order to resemble the actual Pompeii site that we visited for our first on site tests. This allowed us for extra fine tuning and improvement of our simulation and framework, without having to visit numerous times the actual site. With the help of the Superintendence of Pompeii, POMPEII [POM 04], who provided us with all necessary archaeological and historical information, we have selected the 'thermopolium' (tavern) of Vetutius Placidus and we contacted our experiments there. The results are depicted in the figure 5.

8 Conclusion and Future Work

With the current result of our AR Framework, we are able to manage augmented reality full virtual character simulations (body, face and clothes) in real-time cultural heritage environments through a markerless AR tracking system. However, there is still a lot of space for improvement. Firstly we will improve the real-time camera tracking, in order to get higher frame rates in both methods. There is also a lot of work to improve the interaction between the real object and the virtual scene, more accurate registration and automatic set-up phase of the virtual world and the occluders. Finally the 'illumination' registration between the real and the virtual scene is currently been addressed with the introduction of High Dynamic Range Image Based Lighting for virtual character simulations in AR.

Acknowledgements

Special thanks to Tom Molet, Michal Ponder, Sumedha Kshirsagar, Branislav Ulicny, Bruno Herbelin and Rajeev Gupta for their help with the VHD++ framework. The presented work is supported by the Swiss Federal Office for Education and Science and the EU IST FP5 programme, in frame of the EU IST LIFEPLUS project.

References

[ARN 00] Arnold, D.B., Computer Graphics and Archaeology: Realism and Symbiosis, 2000, ACM SIGGRAPH and EUROGRAPHICS: Interpreting the Past, pre-conference proceedings, pp. 10-21.

[MIL 94] Milgram, P., Kishino, F., 1994, A Taxonomy of Mixed Reality Visual Displays, IEICE Trans. Inf. Syst., vol. E77-D, 12, pp 1321-1329.

[AZU 01] Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., MacIntyre, B., 2001, Recent Advances in Augmented Reality, IEEE Computer. Graphics Application, vol. 21, 6, pp 34-47.

[ART 04] ART: Augmented Reality for Therapy, http://mrcas.mpe.ntu.edu.sg/groups/art/

[STI 01] Stricker, D., Dähne, P., Seibert, F., Christou, I., Almeida, L., Ioannidis, N., 2001, Design and Development Issues for Archeoguide: An Augmented Reality based Cultural Heritage On-Site Guide, ICAV3D 2001: Augmented Virtual Environments and 3D Imaging, proceedings, pp. 1-5.

[PAP 02] Papagiannakis, G., Ponder, M., Molet, T., Kshirsagar, S., Cordier, F., Magnenat-Thalmann, N., Thalmann, D., 2002, LIFEPLUS: Revival of life in ancient Pompeii, VSMM2002, invited paper.

[SCH 01] Schwald, B., Figue, J., Chauvineau, E., Vu-Hong, F., 2001, STARMATE: Using Augmented Reality technology for computer guided maintenance of complex mechanical elements, e2001: eBusiness and eWork, proceedings, vol.1, section 1.4.

[WHO 00] Wohlgemuth, W., Triebfürst, G., 2000, ARVIKA: augmented reality for development, production and service, DARE 2000: Designing augmented reality environments, proceedings, pp. 151-152.

[THO 00] Thomas, B., Close, B., Donoghue, J., Squires, J., De Bondi, P., Morris, M., and Piekarski, W., ARQuake: An Outdoor/Indoor Augmented Reality First Person Application, 2000, ISWC 2000: Wearable Computers, proceedings, pp. 139-146.

[GAM 94] Gamma, E., Helm, R., Johnson, R., Vlissides, J., 1994, Design Patterns: Elements of Reusable Object-Oriented Software, ed. Addison-Wesley

[PON 03] Ponder, M., Papagiannakis, G., Molet, T., Magnenat-Thalmann, N., Thalmann, D., 2003, VHD++ Development Framework: Towards Extendible, Component Based VR/AR Simulation Engine Featuring Advanced Virtual Character Technologies, IEEE Computer Society Press, CGI Proceedings, pp. 96-104.

[SAN 99] Sannier, G., Balcisoy, S., Magnenat-Thalmann, N., Thalmann, D., 1999, VHD: A System for Directing Real-Time Virtual Actors, The Visual Computer, ed. Springer, vol.15, 7/8, pp.320-329.

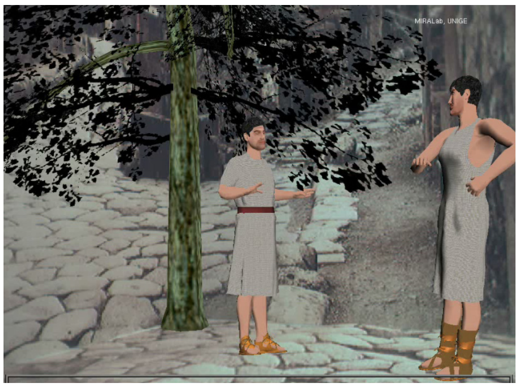
[2D3 04] 2d3: the creator of the Boujou Software Tracker: http://www.2d3.com/.

[POM 04] POMPEII: Archaeological Superintendence of Pompeii: http://www.pompeiisites.org.

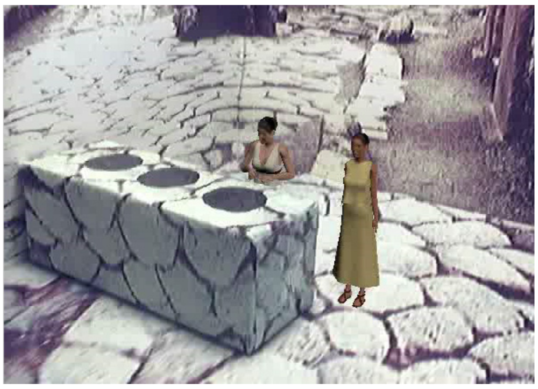
[MAX 04] 3D Studio Max, http://www.discreet.com/products/3dsmax/.

Initial Set-up of 2 mobile workstations on-site (inside the 'thermopolium')

The Pompeian 'thermopolium'



Real-time virtual male Pompeian characters (in the thermopolium maquette)



Real-time virtual Pompeian female characters (with scene occlusions in the thermopolium maquette)

Figure 5: *Mixed Reality offline simulations in various Pompeian residences*

Case-Studies in National and European Research projects

J. Meseth, G. Müller, M. Sattler, R. Sarlette and R. Klein

University of Bonn, Institute for Computer Science II

Abstract

Textile simulation and visualization is an important topic for a large number of companies related to textile design, production and sales all over the world. Research in this area is conducted in many research areas like mathematics, physics and computer science. The wide interest in industry and research is reflected by the large number of projects - national and international - related to this topic. In the following sections, a German and a European research project will be presented exemplarily, including their goals, approaches and solutions.

1. Virtual Try-On

The Virtual Try-On (VTO) [VTO] project (01IRA01A) is a research project funded by the German Minstry of Education and Science (BMB+F) which was started in the beginning of 2001 and ended at July 31st, 2004. The eight partners of the project represent research institutes and industry by equal shares.

The goal of the project was the solution of the problems related to virtual tryon of cloth in a boutique or using an Internet application and to the cost-efficient production of bespoke clothing by simulating the whole process chain from the selection of cloth to the visualization and individual assessment of fit.

In a sense, users should be offered the possibility to use individualized mail-order catalogues with themselves wearing the advertised clothes instead of models. Compared to shopping in stores, this allows the user to choose from a much bigger selection of products in a smaller amount of time. In addition, the user can try on clothes which are currently not available at the store. Compared to current mail-ordering, users can judge the look of clothes depending of their own body measures which leads to higher customer satisfaction and consequently to lower costs for the companies due to decreased returns.

Additionally, the project offers solutions for creation of bespoke clothing based on customer data acquired during the customer virtualization step.

The general conception underlying the project is shown in figure 1. VTO identified two different application scenarios which could both profit from software based on the project: installations in real shops (points of sale (POS)) and Internet applications. While Internet applications offer the user the possibility to select various catalogues containing cloths from several distributers and producers, virtually trying on products from these catalogues and conducting orders, real shops additionally offer the necessary hardware and services for customer virtualization (i.e. body scanners and scanning software) and virtual mirrors (large display devices and customer tracking).

The structure of a system supporting these application scenarios is shown in the right box of figure 1. Scanned data from customers is stored and retrieved from a database. Producers can be queried for available products (which are again stored in a database) which consist of the patterns used for sewing, and the respective materials used for production. The measures of the virtual customer, sewing patterns and material informations are fed into the visualization system, the core part of VTO, where photorealistic visualizations are generated from physics-based simulation results and returned to the virtual shop for display.

While product specific topics like creation and management of Internet applications were targeted in the project as well, this text will concentrate on the research conducted on the visualization part, i.e. dressing of virtualized humans, simulation of physically correct fit and folding, and real-time photorealistic visualization. In the following, results from these parts will be described.

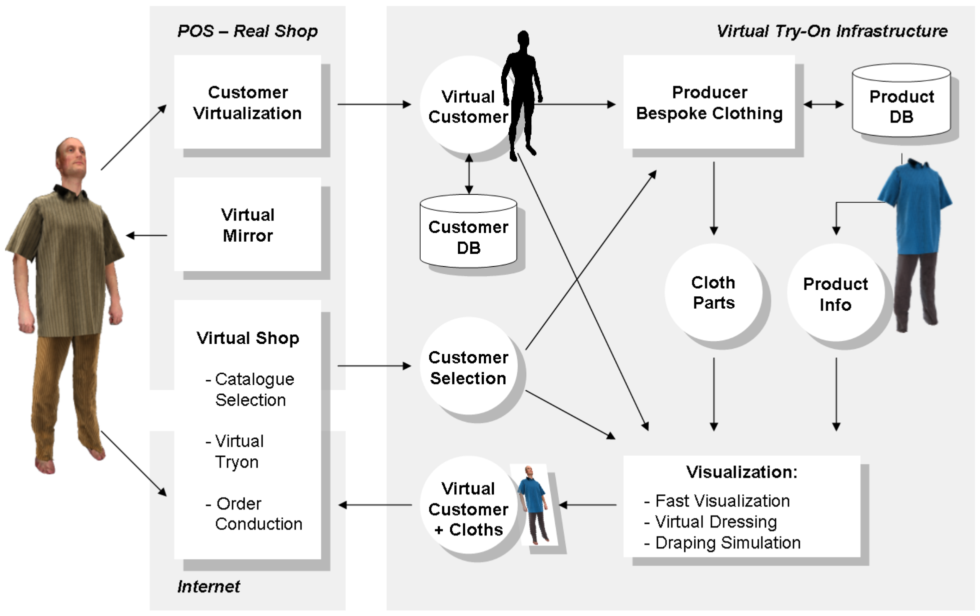


Figure 1: *Underlying structure of the Virtual Try-On project*

1.1. Dressing of Virtual Humans

The first step of cloth visualization after acquisition of human body and cloth data is dressing the body with cloth (i.e. simulation of the process of putting on cloth) to achieve a valid starting point for the cloth simulation. Since a full simulation of the dressing process is very complicated, the project followed a simpler approach utilized in the movie industry [Vis01]: the sewing patterns of a piece of cloth are wrapped around the human body and sewn there.

This approach fits the above structure very well since the highly specialized CAD systems from cloth industry [ass, Lec] store garments as twodimensional geometries of sewing patterns combined with sewing information. Unlike the approaches described by Volino and Magnenat-Thalmann [VMT97, VMT00] and those employed in movie industry which require the sewing patterns to be manually positioned around the body in a time-consuming process before sewing, the approach followed in the project is fully automatic and therefore much faster.

The new approaches developed by Fuhrmann et al. [FGL03] and Groß et al. [GFL03] approximate human body parts by bounding volumes with developable surfaces like cylinders and cones and position the parts on these approximations (see figure 2). The methods pay special attention to avoiding distortions of the garments while positioning them on the approximated avatar since these could lead to false results or even divergence in the following physics-based simulation. In addition, the approximating bounding volumes allow for efficient collision avoidance of the body and the sewing patterns which simplifies the following simulation process drastically (compare [BFA02, BWK03, MKE03] for techniques that resolve collisions during the simulation process). Finally, the methods minimize the distances between designated, corresponding seams to decrease the lengths of simulated distances during the seaming process.

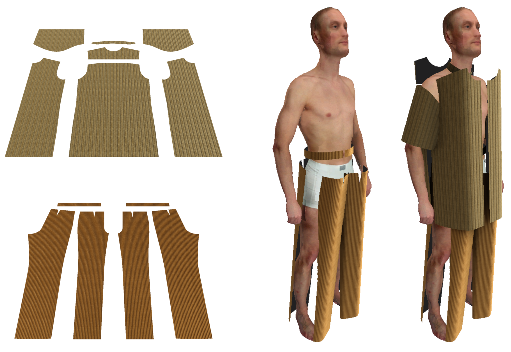


Figure 2: *Automatic prepositioning of the garments represented as sewing patterns (left).*

Employing the new techniques allows simultaneous and fully automatic prepositioning of several garments by utilizing multiple layers of bounding volumes positioned around each other (compare figure 2). This approach allows for easy

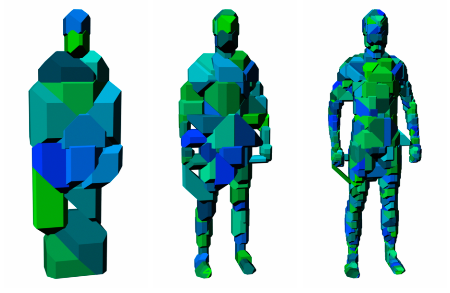


Figure 3: *Various resolution levels of the k-DOP hierarchy.*

definition of dressing orders, i.e. for defining whether a shirt is worn under or over the pants.

1.2. Physics-Based Simulation of Folding

Following the automatic prepositioning step the physics-based simulation of clothes is executed. The developed models incorporate measured material properties and target as realistic as possible simulation of the dynamics of cloth to provide the customer with an as realistic as possible impression of the fit of the garment.

The model for physics-based textile simulation developed by Etzmuß et al. [EKS03] is based on a finite-element discretization of the continuous elasticity equation. The visco-elastic behavior is approximated based on a linear description of distortion which allows for definition of highly time-efficient methods for integration over time. Since the linear formulation yields valid results for small translations only and since such assumptions can generally not be made for textiles (concerning bending, textiles have low resistance with respect to shear and stretch-forces which can result in large deformations), every finite element is assigned its own local reference frame within which the inner forces are computed and which is rotated with the finite element. Intergration over time is fianlly done using an implicit Euler method which requires to solve a sparse system of linear equations for every time step. For very detailed geometric models of textiles this method can be parallelized efficiently [KB04].

The abovementioned method allows for modelling of measured material properties like stretch forces in warp- and fill-direction, shear forces, bend forces in warp- and fill-direction and lateral contraction. For every simulated material such visco-elastic parameters are extracted from the hysteresis curves measured by the Kawabata measuring system [Kaw80]. External forces like resistance to wind and effects of air flows can be integrated into the model [KKTW04] and increase the realistic appearance of the folding computed by the simulation.

Since the simulator solves the dynamic elasticity equation it can handle scenes including moving objects. This allows

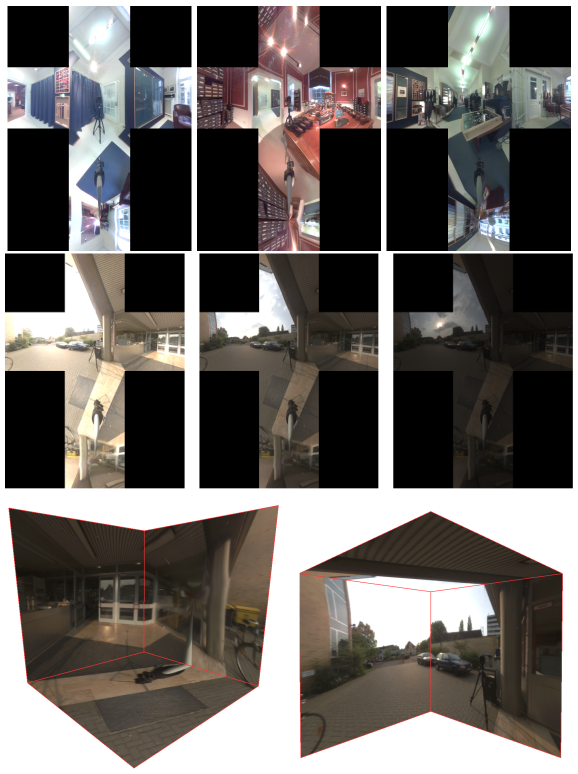


Figure 4: *Acquired HDR lighting environments. Top row: Acquired HDR shop environments. Middle row: Entrance of the University of Bonn for different exposure times. Bottom row: Cut, cubic environment map.*

animations of the avatar based on movement data (e.g. from a motion capture system) and the behavior of the textiles it is dressed with. In practice this allows customers to better predict the fit of selected cloths.

The simulation system employed in the Virtual Try-On project implements efficient collision-detection and collision-answer methods – one of the big challenges of simulation systems – by employing hierarchies of *k*-DOP bounding volumes (see figure 3) to speed up the intrinsic quadratic complexity of collision detection. These hierarchies are traversed and contained objects are checked for collision which allows fast and efficient identification of large parts of the scene which are not colliding. Due to potentially changing since animated scenes these hierarchies have to be changed frequently. Fortunately the update costs can be minimized combining efficient bottom-up traversals with temporal coherence among multiple time steps [MKE03].

After all colliding primitives were detected, the collision-answer has to restore a valid, collision-free state. The Virtual Try-On simulator employs two different methods.

- The movement of individual surfaces into special direc-

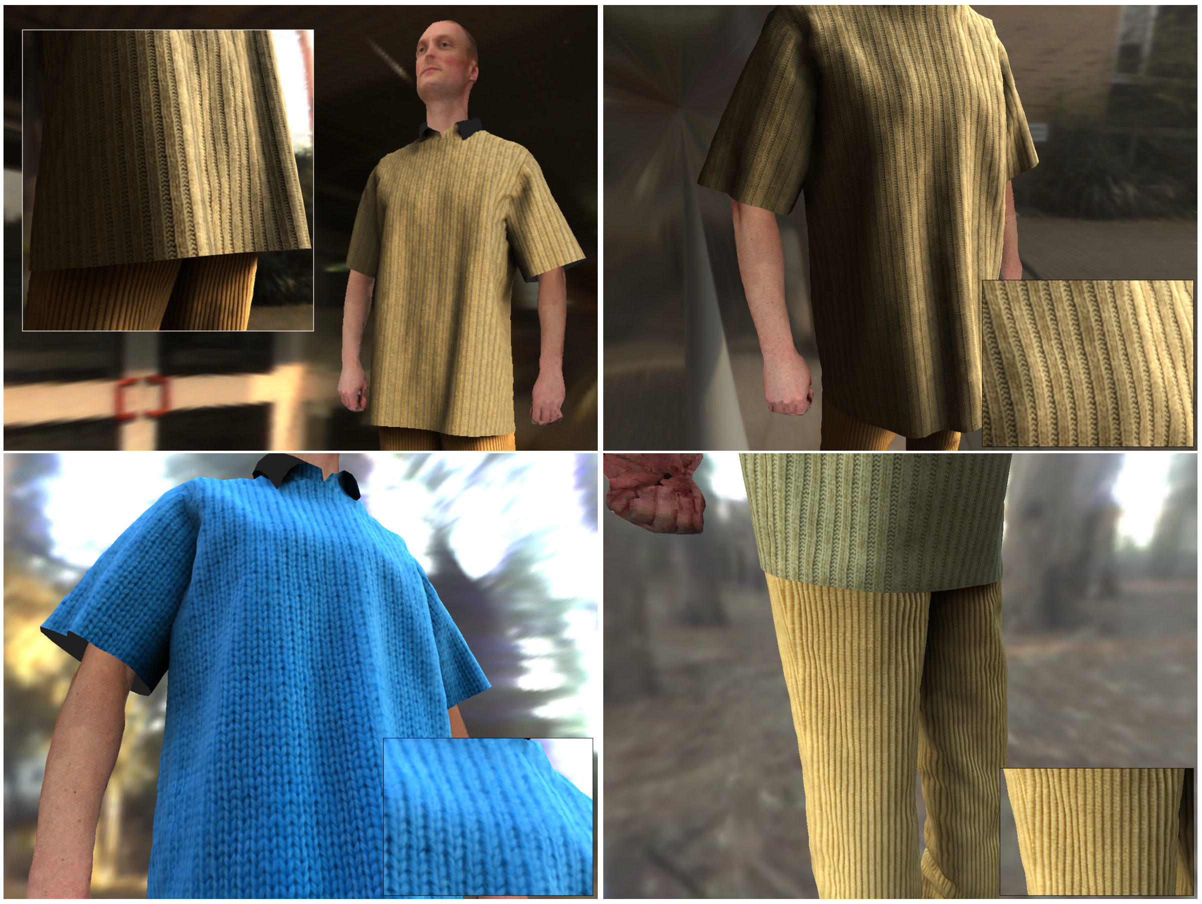


Figure 5: *Photorealistic visualization results from the Virtual Try-On project.*

tions can be restricted by several conditions in the differential equation solver which prevents further penetration of colliding objects. Already penetrating vertices are moved to the surface of the penetrated object.
- Repelling force potentials resolve self-collisions by making the colliding objects stride away from each other.

The simulator's output are three-dimensional meshes representing the geometry of the cloth which are handed to the photorealistic visualization step.

1.3. Photorealistic Visualization

The final step of the visualization process in the Virtual Try-On project is the photorealistic, three-dimensional visualization of the clothes selected by the customer. Special attention is payed to reproduce the correct appearance of cloth by accounting for the physically correct reflection properties to provide the customer with a correct impression (look and feel) of the textile.

These properties were acquired as BTFs by the setup described in Sattler et al. [SSK03] which was described in detail in section 2 of the 'Acquisition and Real-Time Rendering of Textile' part as well. In addition, the lighting environments of several potential points of sales were acquired as HDR environments [Deb98] (see figure 4).

To generate high-quality visualizations of the results of the cloth simulation step, the resulting geometry models are BTF textured with the acquired data. Since the models are rather small and since high-frequency shadows play an important role at least in some of the acquired lighting environments, a new BTF rendering algorithm was developed [SSK03] and combined with visibility maps, which allow accurate, high-frequency shadows at the expense of substantial precomputation time whenever the lighting situation changes. Since the dressed avatar should be able to be rotated in the lighting environment, rotations of the lighting environment are precomputed and encoded using principal component analysis (PCA).

Results of the rendering algorithm are presented to the customer either using virtual mirrors in the real shops or on a monitor screen in Internet applications. Example results are shown in figure 5 and the accompanying videos.

2. RealReflect

Virtual Reality (VR) applications like Virtual Prototyping try to convey as realistic scenarios to the users as possible. The imagery shown on special output devices like head-mounted displays, caves or projection walls is usually of rather low quality, especially with respect to physical realism. Reasons are the use of only very simple material representations like lit or bump-mapped textures and the inability to visualize global illumination results other than from radiosity algorithms in real-time. As a result, in such systems only the shape of objects can be judged correctly, not their overall appearance. Also, the atmosphere and impression of the light distribution in interiors cannot be rendered convincingly. As a consequence, not even radiosity-based VR systems permit the verification of safety regulations which strongly depend on specular reflections. Examples of such scenes, which are almost impossible to display convincingly using current VR technology, are the interiors of cars and buildings. Various other limitations limit the applicability of current VR systems even further.

The target of the RealReflect [Rea] project (IST-2001-34744) – a project funded by the European Union comprising nine partners from universities and industry that started in April 2002 – is to overcome these limitations by employing new techniques that get incorporated into a new, high-quality acquisition, processing and rendering pipeline. Realistic materials based on BTFs replace the simple materials, accurate texture synthesis and texture mapping algorithms help applying these materials to the rendered objects. Physically more accurate light simulation by photon-tracing replaces the radiosity computations. The final rendering of results is achieved by Surface Light Field (SLF) rendering methods that highly depend on optimized level-of-detail (LOD) representations in order to achieve realistic results in real-time. Completely unknown to current VR and standard cloth visualization systems, real-time tone mapping adjusts the colors of the rendered images to the visual properties of both the human visual system (HVS) and the properties of the display device. Other than most cloth simulation and visualization projects, RealReflect assumes static geometry models and therefore requires no physics-based simulation of cloth.

Although various research results were presented for most of the tasks of the RealReflect project, three main challenges remain within the project.

- First, other than in standard cloth visualization systems, the scenes faced in the project are huge. While car models typically contain millions of triangles or tens of thousands of parametric surfaces, architectural models may contain hundreds of millions of polygons. Rendering of these scenes requires out-of-core algorithms due to the sheer amount of data – both geometry and material – required to represent them.
- Second, the final rendering has to be done in real-time. While out-of-core rendering is a challenging problem by itself, real-time rendering of large scenes is even more demanding.
- Third and most important for this tutorial, the rendered results need to be high quality including both local and global lighting effects. In order to visualize the results of the high quality lighting simulation as realistically as possible, the whole new rendering pipeline requires high-quality techniques that are perfectly tuned to fit the needs of each other.

In the following subsections, first the acquisition, processing and rendering pipeline of the RealReflect project will be introduced. Afterwards selected stages will be described in further details.

2.1. Rendering Pipeline

The final goal of the RealReflect project is to set up a new high-quality acquisition, processing and rendering pipeline for VR systems. Figure 6 depicts the pipeline and its stages which lead from acquisition of materials and light sources over texturing (synthesis and mapping) to LOD and occlusion precomputation, global illumination precomputation, interactive rendering and finally real-time tone-mapping.

The inputs to the pipeline are real, rather small material probes, real light sources and geometry files. The first stages digitize the real input by automatically measuring reflectance properties of materials and light source properties. Next, the digitized materials are analyzed, compressed and texture synthesis algorithms are applied to generate arbitrarily large material textures. At the same time, the input geometry is parameterized to generate appropriate texture coordinates. The resulting textured models are simplified, culling hierarchies are generated, and the outputs are stored in a scene file. The stored scenes can afterwards be previewed using the BTF renderer, which utilizes approximations of the real light sources which are computed in the Light Source Estimation stage. Alternatively, the light distribution of the scene can be computed by the High Quality Light Simulation module and afterwards be rendered by the SLF renderer. Output images from both types of renderers finally get tone-mapped in real-time to yield the final output images.

The following sections describe selected stages in further detail and provide details on already existing results achieved within the project. For more details concerning these and the other stages we refer to [KMM*03].

2.2. Data Acquisition

In the RealReflect project, data acquisition refers to the automatic acquisition of materials and luminaires (light sources).

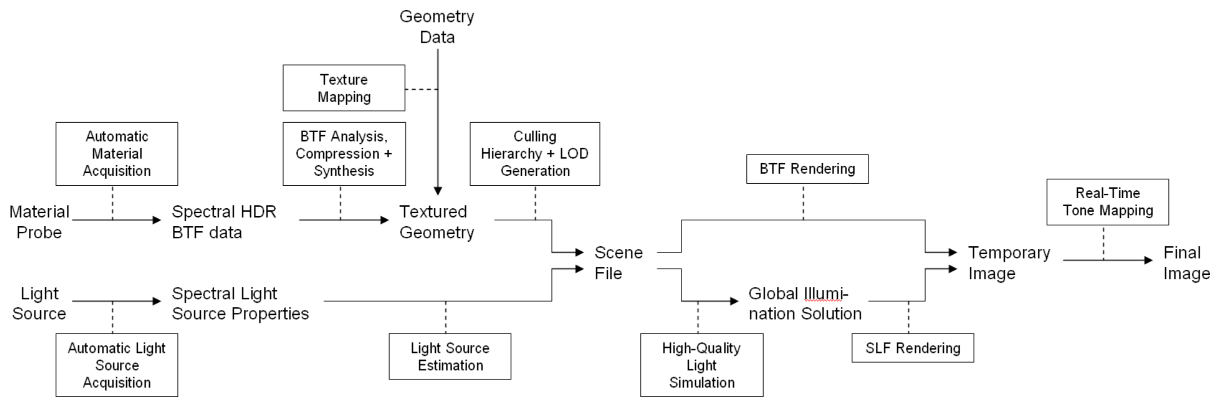

Geometry Data
Texture Mapping
Automatic Material Acquisition
BTF Analysis, Compression + Synthesis
Culling Hierarchy + LOD Generation
BTF Rendering
Real-Time Tone Mapping
Material Probe
Spectral HDR BTF data
Textured Geometry
Scene File
Temporary Image
Final Image
Light Source
Spectral Light Source Properties
Global Illumination Solution
Automatic Light Source Acquisition
Light Source Estimation
High-Quality Light Simulation
SLF Rendering

Figure 6: *The RealReflect rendering pipeline*

The acquisition of geometry models for virtual prototyping is typically a task of designers and therefore out of the scope of the project.

Reflectance properties of materials – represented as BTFs – are acquired as described in section 2 of the 'Acquisition and Real-Time Rendering of Textile' part. Unlike the material samples stored in the material database mentioned above, RealReflect works with material samples of at least 512×512 pixels and utilizes high dynamic range color information in contrast to the 8 bit RGB BTFs stored in the online database.

The automatic acquisition of light emission properties of real-world luminaires is a necessary goal of the RealReflect project since this data on the one hand is necessary for well-funded safety relevant decisions and on the other hand usually is considered an intellectual property of the manufacturer and is thus not published.

Although few publications cover this topic, the recent publication by Goesele et al. [GGHS03], which is the basis for light acquisition in the RealReflect project, reflects the still lasting interest in this topic. After conversion of the emission properties, these are converted to the IESNA [Ill02] file format and made available for rendering.

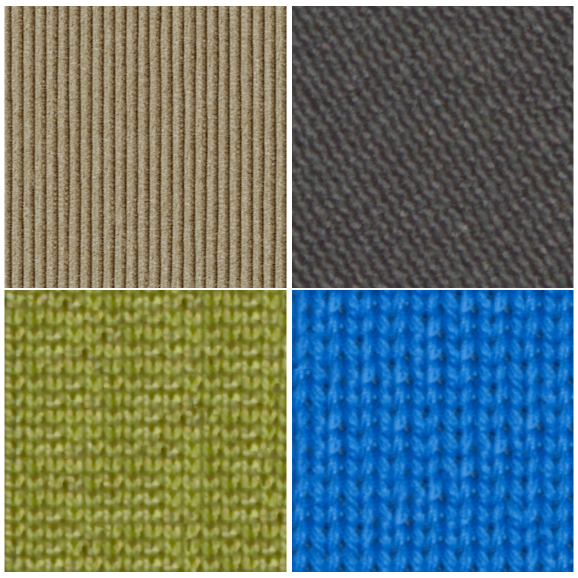


Figure 8: *Experimental results from BTF synthesis of Corduroy, a cushion cover from car industry and two kinds of knitted wool. Images courtesy of the University of Bonn.*

2.3. Texture Synthesis

Texture synthesis denotes the process of generating a large texture with desired appearance attributes (i.e. structure, color, etc.) but without obvious repetitions, which is inevitable for visualizing models with large, textured areas. Texture synthesis is an especially indispensable part of the RealReflect project (even more than for other visualization projects related to cloth), since the measured material samples have a size of 10×10 cm (and are typically not repeatable), whereas the rendered models have surface areas of several square meters.

The final goal of the project is the development of parametric models for both color and BTF textures which tremendously reduces the storage costs for textures compared to digitized ones – especially helpful in the case of otherwise memory-intense BTF textures – by storing a small number of parameters only.

For the case of color textures, such models, based on the Markov Random Field (MRF) method, were developed by the project partners already [Hai02, Gri02, HH02a, HH02b]. The main challenge of BTF synthesis is the high dimensionality of BTFs compared to low-dimensional standard textures.

As an alternative, BTF synthesis algorithms which are not

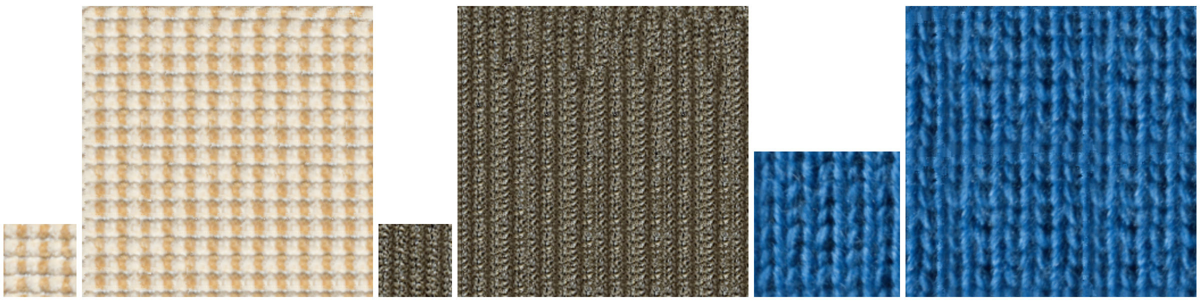


Figure 7: *Results from texture synthesis (left: Proposte, middle: corduroy, right: knitted wool). The small images represent exemplary views of the samples, the large images the according views from the synthesized images. For Proposte and knitted wool, the structure of the synthesized image closely resembles the original, for corduroy the random component of the algorithm destroys the original structure.*

parameter based and therefore require far more storage are used and developed in the project. As a first step, a modified version [MMK03a] of the approach of Tong et al. [TZL*02] was employed. Figure 7 shows that this approach produces good results for several BTFs (e.g. the highly structured Proposte material) but yields insufficient quality for others (e.g. Corduroy). Therefore experiments with new BTF synthesis methods not published yet were conducted and lead to very pleasing results (see figure 8). Nevertheless, an optimal approach was not determined yet.

2.4. Texture Mapping

Texture mapping, i.e. the process of applying a texture to a surface by assigning texture coordinates, is another key application in the RealReflect pipeline. Other than for cloth production, the geometry used in virtual prototyping usually directly originates from CAD systems which typically either contain no texture coordinates at all or the texture coordinates resulted from simple planar projections. These coordinates neither minimize texture stretch (necessary for a uniform coverage of the surface) nor do they allow arbitrary texture orientation (reflecting real-world structure).

To generate such parametrizations for triangular models, a new, fast algorithm was developed by Degener et al. [DMK03] which guarantees avoidance of face flips while allowing to balance the amount of angle and area preservation of the parameterization method by a single, intuitive parameter without requiring fixed parameterization boundaries. Figure 9 shows the gear box of a car textured with this method.

As an alternative, texture mapping can be done by directly synthesizing BTFs on the surface of the model. The main advantages of this kind of synthesis are applicability to arbitrary manifold surfaces and memory efficiency. The first advantage comes from local parameterization while the latter holds for patch-based methods which simply generate texture coordinates. One recent approach was presented by Magda and Kriegman [MK03] but is unfortunately limited

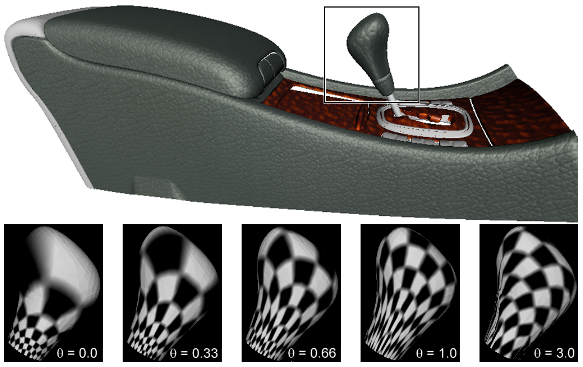


Figure 9: *Top: BTF textured gear box model (courtesy of DaimlerChrysler) based on the computed parameterization. Bottom: the effect of varying θ is shown: for $\theta = 0$ a conformal mapping results, for $\theta = 1$ angle and area preservation are equally important. With growing θ area preservation improves.*

to standard textures yet. Extensions of this method to BTFs are to be developed.

2.5. High Quality Light Simulation

Virtual Prototyping applications that want to evaluate the appearance of complex scenes at any scale need to take global illumination effects into account. In current VR systems, solutions from radiosity algorithms are used for this task, which unfortunately entails some drawbacks.

- Radiosity algorithms assume diffuse environments. Unfortunately, especially for safety relevant evaluations of cars the specular reflections in the windshield are crucial, which cannot be captured by radiosity based algorithms.
- Radiosity algorithms require a subdivision of the scene into patches. In order to achieve approximately correct

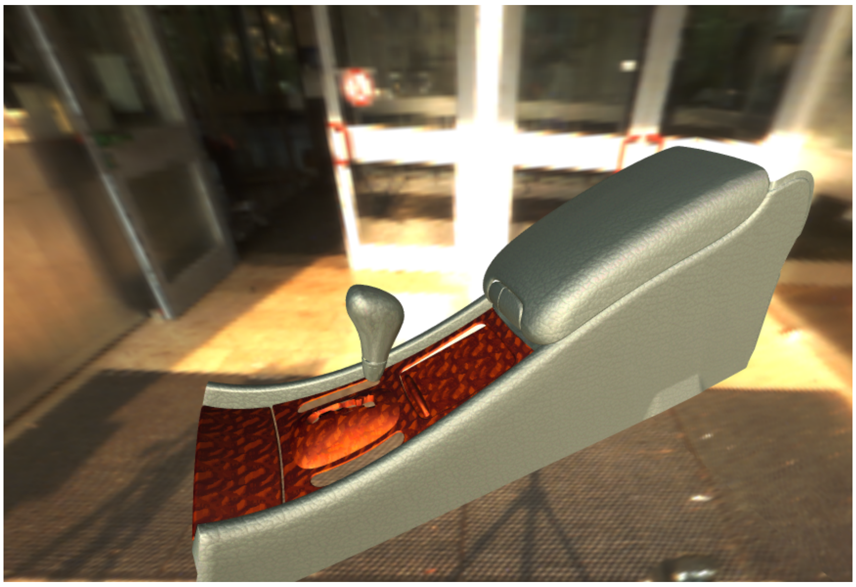


Figure 10: *Parts of the gear box model (courtesy of DaimlerChrysler) covered with synthetic leather and lacquered mahagony BTFs.*

lighting results for interiors with arbitrarily complex material properties, the number of patches needs to be very high, resulting in unpractically long computation times.

The RealReflect project therefore suggest to use stochastic particle tracing (namely photon-mapping [Jen02]) in combination with ray-tracing methods to compute realistic, high-quality global illuminations for static scenes. Since this obviously requires a preprocessing step, the solution is stored as Surface Light Fields (SLF) and rendered by a SLF renderer in real-time. Unfortunately, this prohibits changes to the scene. Approaches like Selective Photon Tracing [DBMS02] will help to reduce update times for small changes which might occur during virtual prototyping processes.

A completely new and challenging task of the RealReflect project is the combination of ray-tracing with accurate, measured material representations which has never been done before.

2.6. High-Quality Rendering

The BTF rendering is supposed to offer a high-quality preview mechanism for the scenes before the actual global illumination solution is computed. BTF rendered scenes may contain most important, low-frequency lighting effects via precomputed radiance computation but miss high-frequency, often safetly relevant features like reflections in the windshield of a car.

The employed methods for real-time visualization [MMK03b, Sch04, MMK04] were thoroughly presented in section 4 in the 'Acquisition and Real-Time Rendering of Textile' part. Figure 10 shows parts of the gear-box covered with synthetic leather and lacquered mahagony BTFs. If utilized in VR environments, such visualizations tremendously increase the realism.

In addition to the BTF rendering, the Surface Light Field rendering algorithms incorporate all global effects simulated by the high quality light simulation.

2.7. Tone Mapping

The final step of the RealReflect rendering pipeline is tone mapping, which denotes the task of mapping a high-dynamic range image (here represented as floating-point valued RGB values) to a low-dynamic range image which can be displayed on the output device (e.g. 8 bit RGB values for standard monitors). This mapping should be optimized for the human visual system (HVS), i.e. the perception of humans should be taken into account.

Although omitted in nearly all existing visualization systems, tone mapping is an important stage of the RealReflect pipeline since both BTFs and results from the global illumination solver are required to include HDR color values to correctly simulate reality. Simply clamping or incorrectly mapping these values to the low dynamic output luminance range would prohibit realistic judgement of reflectance behavior and especially prohibit safety relevant decisions to be made.

Within the project a new local tone mapping operator, which can efficiently be executed in graphics hardware, was developed by Drago et al. [DMAC03] based on an evaluation of existing tone mapping operators [DMMS02]. Since the algorithm requires knowledge about global image features, Artusi et al. [ABWW03] suggested a framework which can deliver at least interactive performance to nearly all existing local tone-mapping operators. Further developments in the project include time-dependent tone mapping operators.

2.8. VR System

Many of the stages of the acquisition, processing and rendering pipeline will be integrated into IDO:Base, the VR system of IC:IDO. The target of the project is that this system will be used in many styling and design applications both in the automotive industry and architecture in the new future.

Acknowledgements

Many works related to this tutorial were at least partially funded by the Virtual Try-On or RealReflect project. We thank our project partners from Virtual Try-On

- HUMAN SOLUTIONS GmbH, Germany,
- WSI/GRIS, University of Tübingen, Germany
- Fraunhofer Institute for Computer Graphics, Germany,
- Hohenstein Institute for Clothing Physiology,
- German Research Center for Artificial Intelligence GmbH, Germany,
- Odermark, Germany, and
- Cove&Co, Germany

and RealReflect

- Computer Graphics Group of the Vienna Technical University, Austria,
- Max-Planck Institute for Computer Science, Saarbrücken, Germany,
- Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic (UTIA), Czech Republic,
- French National Institute for Research in Computer Science and Control (INRIA), Grenoble, France,
- DaimlerChrysler AG, Germany,
- IC:IDO GmbH, Germany,
- Faurecia, France, and
- virtual reality architects, Austria

for the excellent collaboration.

The gear box model was kindly provided by the DaimlerChrysler AG.

References

[ABWW03] ARTUSI A., BITTNER J., WIMMER M., WILKIE A.: Delivering Interactivity to Complex Tone Mapping Operators. In *Eurographics Symposium on Rendering 2003* (Leuven, Belgium, June 2003). 9

[ass] assyst/bullmer intelligent solutions. http://www.assyst-intl.com. 2

[BFA02] BRIDSON R., FEDKIW R. P., ANDERSON J.: Robust Treatment of Collisions, Contact, and Friction for Cloth Animation. In *Proceedings of SIGGRAPH 2002* (2002), pp. 99–104. 2

[BWK03] BARAFF D., WITKIN A., KASS M.: Untangling Cloth. *Transactions on Graphics 22*, 3 (2003), 862–870. 2

[DBMS02] DMITRIEV K., BRABEC S., MYSZKOWSKI K., SEIDEL H.-P.: Interactive Global Illumination Using Selective Photon Tracing. In *Eurographics Workshop on Rendering 2002* (2002), pp. 21–34. 8

[Deb98] DEBEVEC P.: Rendering Synthetis Objects into Real Scenes: Bridging Traditional and Image-Based Graphics with Global Illumination and High Dynamic Range Photography. In *Proceedings of SIGGRAPH 1998* (1998), pp. 189–198. 4

[DMAC03] DRAGO F., MYSZKOWSKI K., ANNEN T., CHIBA N.: Adaptive Logarithmic Mapping For Displaying High Contrast Scenes. In *Proceedings of Eurographics 2003* (September 2003). 9

[DMK03] DEGENER P., MESETH J., KLEIN R.: An Adaptable Surface Parameterization Method. In *12th International Meshing Roundtable* (2003). 7

[DMMS02] DRAGO F., MARTENS W., MYSZKOWSKI K., SEIDEL H.-P.: *Perceptual Evaluation of Tone Mapping Operators with Regard to Similarity and Preference*. Research Report MPI-I-2002-4-002, Max-Planck-Institut für Informatik, 2002. 9

[EKS03] ETZMUSS O., KECKEISEN M., STRASSER W.: A Fast Finite Element Solution for Cloth Modelling. In *Proceedings of Pacific Graphics* (2003). 3

[FGL03] FUHRMANN A., GROSS C., LUCKAS V.: Interaction-Free Dressing of Virtual Humans. *Computers and Graphics 27*, 1 (2003), 71–82. 2

[GFL03] GROSS C., FUHRMANN A., LUCKAS V.: Automatic Pre-Positioning of Virtual Clothing. In *Proceedings of the Spring Conference on Computer Graphics* (2003), pp. 113–122. 2

[GGHS03] GOESELE M., GRANIER X., HEIDRICH W., SEIDEL H.-P.: Accurate Light Source Acquisition and Rendering. In *Proceedings of*

SIGGRAPH 2003 (San Diego, California, July 2003), pp. 621–630. 6

[Gri02] GRIM J.: A Discrete Mixtures Colour Texture Model. In *Proceedings Texture 2002* (2002), Chantler M., (Ed.), pp. 59–62. 6

[Hai02] HAINDL M.: Model-Based Pattern Recognition. In *Pattern Recognition and String Matching*. Kluwer Academic Publishers, December 2002. 6

[HH02a] HAINDL M., HAVLÍČEK V.: A Multiscale Colour Texture Model. In *Proceedings of the 16th IAPR International Conference on Pattern Recognition* (2002), vol. I, pp. 255–258. 6

[HH02b] HAINDL M., HAVLÍČEK V.: A Simple Multispectral Multiresolution Markov Texture Model. In *Proceedings Texture 2002* (2002), Chantler M., (Ed.), pp. 63–66. 6

[Ill02] ILLUMINATING ENGINEERING SOCIETY OF NORTH AMERICA: *BSR/IESNA Publication LM-63-2002*. www.iesna.org, 2002. 6

[Jen02] JENSEN H. W.: A Practical Guide to Global Illumination using Photon Mapping. In *Siggraph 2002 Course Notes* (July 2002). 8

[Kaw80] KAWABATA S.: *The Standardization and Analysis of Hand Evaluation*. The Textile Machinery and Society of Japan, Osaka, 1980. 3

[KB04] KECKEISEN M., BLOCHINGER W.: Parallel Implicit Integration for Cloth Animations on Distributed Memory Architectures. In *Eurographics Symposium on Parallel Graphics and Visualization* (2004). 3

[KKTW04] KECKEISEN M., KIMMERLE S., THOMASZEWSKI B., WACKER M.: Modelling Effects of Wind Fields in Cloth Simulation. *Journal of WSCG 12*, 2 (2004), 205–212. 3

[KMM*03] KLEIN R., MESETH J., MÜLLER G., SARLETTE R., GUTHE M., BALÁZS Á.: RealReflect: Real-time Visualization of Complex Reflectance Behaviour in Virtual Prototyping. In *Proceedings of Eurographics 2003 Industrial and Project Presentations* (2003). 5

[Lec] Lectra. http://www.lectra.com. 2

[MK03] MAGDA S., KRIEGMAN D.: Texture Synthesis on Arbitrary Meshes. In *Proceedings of the Eurographics Symposium on Rendering* (2003), pp. 82–89. 7

[MKE03] MEZGER J., KIMMERLE S., ETZMUSS O.: Hierarchical Techniques in Collision Detection for Cloth Animation. *Journal of WSCG 11*, 2 (2003), 322–329. 2, 3

[MMK03a] MESETH J., MÜLLER G., KLEIN R.: Preserving Realism in real-time Rendering of Bidirectional Texture Functions. In *OpenSG Symposium 2003* (April 2003), Eurographics Association, Switzerland, pp. 89–96. 7

[MMK03b] MÜLLER G., MESETH J., KLEIN R.: Compression and real-time Rendering of Measured BTFs using local PCA. In *Vision, Modeling and Visualisation 2003* (November 2003), pp. 271–280. 8

[MMK04] MÜLLER G., MESETH J., KLEIN R.: Fast Environmental Lighting for Local-PCA Encoded BTFs. In *Computer Graphics International 2004 (CGI2004)* (June 2004). 8

[Rea] RealReflect. http://www.realreflect.org. 5

[Sch04] SCHNEIDER M.: Real-Time BTF Rendering. In *Proceedings of CESCG 2004* (2004). 8

[SSK03] SATTLER M., SARLETTE R., KLEIN R.: Efficient and Realistic Visualization of Cloth. In *Proceedings of Eurographics Symposium on Rendering 2003* (Leuven, Belgium, June 2003). 4

[TZL*02] TONG X., ZHANG J., LIU L., WANG X., GUO B., SHUM H.-Y.: Synthesis of Bidirectional Texture Functions on Arbitrary Surfaces. In *Proceedings of SIGGRAPH 2002* (2002), pp. 665–672. 7

[Vis01] VISUAL EFFECTS SOCIETY: *From Ivory Tower to Silver Screen*. SIGGRAPH 2001 Course Notes, 36, 2001. 2

[VMT97] VOLINO P., MAGNENAT-THALMANN N.: Developing Simulation Techniques for an Interactive Clothing System. In *Proceedings of Virtual Systems and MultiMedia '97* (1997), pp. 109–118. 2

[VMT00] VOLINO P., MAGNENAT-THALMANN N.: *Virtual Clothing. Theory and Practice*. Springer, 2000. 2

[VTO] Virtual Try-On. http://www.virtualtryon.de. 1