Collision Detection and Post-Processing for Physical Cloth Simulation

Dissertation
der Fakultät für Informations- und Kognitionswissenschaften
der Eberhard-Karls-Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Dipl.-Phys. Stefan Kimmerle
aus Nürtingen

Tübingen
2005

Tag der mündlichen Qualifikation:	29.06.2005
Dekan:	Prof. Dr. M. Diehl
1. Berichterstatter:	Prof. Dr.-Ing. Dr.-Ing. E.h. W. Straßer
2. Berichterstatter:	Prof. Dr.-Ing. M. Teschner (Albert-Ludwigs-Universität Freiburg)

To my mother
who taught me
never to give up

Zusammenfassung

Um deformierbare Materialien wie Textilien oder menschliches Gewebe physikalisch simulieren zu können, muss eine Reihe komplexer Probleme gelöst werden. Diese Arbeit befasst sich dabei im Speziellen mit der Detektion und Behandlung von auftretenden Kollisionen sowie der Nachbearbeitung simulierter Netze zur Steigerung der visuellen Qualität. Dabei werden Anwendungen aus dem Bereich simulierter Textilien und der virtuellen Anprobe von Kleidung vorgestellt. Insbesondere wird auf das am WSI/GRIS entwickelte Textilsimulationssystem TüTex eingegangen.

Zunächst wird der State of the Art zur Kollisionsdetektion deformierbarer Objekte vorgestellt, und es wird ausführlich diskutiert, welches Verfahren am besten für welche Anwendung geeignet ist. Für die Anforderungen, die TüTex an die Kollisionsdetektion stellt, wird gezeigt, dass sich dafür besonders Bounding-Volume-Hierarchien eignen. Diese werden im Weiteren mit einem stochastischen Sampling zu einer neuen Kollisionsdetektionsmethode kombiniert. Dieses neue Verfahren erlaubt eine Abwägung zwischen Geschwindigkeit und Qualität der Detektion und erhöht damit auch deutlich ihre Performance. Im Folgenden wird eine Impuls-basierte Methode zur Auflösung komplexer Kollisionen und Selbstkollisionen vorgestellt, die sowohl für statische als auch dynamische Kollisionsobjekte stabile Simulationen sicherstellt.

Da Textilsimulationen mit hoch aufgelösten Netzen nach wie vor sehr zeitintensiv sind, wird vorgeschlagen, grobe Netze zu simulieren und diese anschließend geometrisch nach zu bearbeiten. Dazu trägt diese Arbeit zwei Verfahren bei. Um die sichtbare polygonale Struktur von groben Netzen zu beseitigen werden Subdivisionmethoden benutzt. Dabei werden interpolierende und approximieren-

de Verfahren in Bezug auf die Eignung bei virtuellen Textilien verglichen. Da der Subdivisionschritt selbst auch wieder zu Kollisionen vor allem zwischen dem modifizierten Textilnetz und seiner Umgebung führen kann, werden diese Verfahren mit einer kontinuierlichen Kollisionsdetektion und Kollisionsantwort kombiniert. Als zweites Verfahren zur Nachbehandlung virtueller Textilien werden Faltentexturen vorgeschlagen. Da grobe Netze keine feinen Falten modellieren können, werden diese Details durch Texturen hinzugefügt. Diese Texturen werden dabei auf Basis der Deformation der Netze generiert und können als Bump- oder Displacement-Map verwendet werden. Im Gegensatz zu früheren Verfahren wird diese Faltentextur ohne Benutzerinteraktion generiert, was die Verwendung in automatischen Simulationssystemen wie TüTex ermöglicht. Des Weiteren werden in dieser Arbeit erstmals diese Texturen mit der kontinuierlichen Kollisionsdetektion kombiniert, um ein kollisionsfreies Displacement-Mapping zu realisieren. Beide Nachbearbeitungsverfahren führen zusammen mit groben Netzen zu visuell vergleichbaren Ergebnissen wie die Simulation hoch aufgelöster Netze ohne Nachbearbeitung, ermöglichen aber deutlich kürzere Simulationszeiten.

Die in dieser Arbeit entwickelten Konzepte wurden in mehrere Systeme zur Kleidersimulation integriert. Mit Virtual Try-On wurde dabei das erste System umgesetzt, welches die physikalisch-basierte Simulation von Kleidung basierend auf CAD-Schnittmustern, physikalischen Materialparametern und 3D-Körperscans ermöglicht. Außerdem wurde der Textilsimulator TüTex als Plugin für die Modellierungssoftware Alias Maya weiter entwickelt, um über eine effiziente und komfortable Test- und Visualisierungsumgebung zu verfügen.

Acknowledgements

This thesis was conducted during my occupation as Research Assistant at the Graphical-Interactive Systems group of the Wilhelm-Schickard-Institute (WSI/ GRIS), University of Tübingen as well as during my research stays at GRAVIR/ IMAG, INRIA Rhône-Alpes, Grenoble[1].

First and foremost, I would like to thank Prof. Dr.-Ing. Dr.-Ing. E.h. Wolfgang Straßer for the opportunity to work on this dissertation, and his constant support and advice. I also would like to thank Prof. Dr.-Ing. Matthias Teschner who agreed to be part of my graduation committee. Furthermore I am very grateful to Prof. Dr. Marie-Paule Cani and Prof. Dr. François Faure for their hospitality and the many discussions during my stays in Grenoble.

Special thank goes to all my colleagues, without whom this work would not have been possible, especially to Dr. Olaf Etzmuß, Michael Keckeisen, Johannes Mezger, and Prof. Dr. Markus Wacker. Moreover, I thank the students Egon Bachmann, Martin Gruber, Christian Holzer, Christian Michel, Matthieu Nesme, Simon Pabst, and Bernhard Thomaszewski, who with their work during student and diploma theses delivered valuable contributions to the implementation of the presented algorithms.

Last but not least, I would like to express my gratitude for all kinds of invaluable support and experiences in life to my family as well as to my wife Simone, who patiently bore with me during the numerous nights and weekends I worked through during the last years.

[1]This work was partially supported by a DAAD scholarship for a stay at INRIA Rhône-Alpes in Grenoble, France, and by a grant within the bmb+f *Virtual Try-On* project.

CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Cloth Simulation and Virtual Try-On

Clothing is part of everyone's life and the subject of a huge industry of designers, manufacturers and retailers. Everybody has an intuitive knowledge of the dynamic behavior of cloth and can visually distinguish various fabrics. Consequently the authentic and exact modeling of virtual clothing is a key aspect in the creation of virtual humans, a major goal of computer graphics. In this context, the manual modeling of virtual clothing is a very tedious process and becomes infeasible for animated scenes. Hence, the physically-based simulation of clothing is used to dress virtual humans with garments that show a realistic draping behavior with natural folds and wrinkles. In combination with virtual humans, virtual cloth is employed in a large scope for various purposes. While in the entertainment industry cloth simulations are mainly used to increase the realism of movies or video games, the application scenarios in the textile industry also yield in other directions.

In recent years three main scenarios have evolved in this area. In the first application, a virtual fitting room allows customers the selection, try-on and virtual evaluation of clothing in combination with the customer's virtual counterpart, the

so-called avatar. This virtual fitting room clearly simplifies sales over the Internet and also eases the sales in real boutiques. The second scenario where textile simulations are applied is the design process of garment. With sophisticated cloth simulation software it is possible to facilitate the design process, allow rapid-prototyping, and even make interactive design and tailoring in 3D possible. The third scenario is a combination between rapid-prototyping and the animation of motions where the manufacturers of clothing, e.g. sports clothing, can test the fit and dynamic behavior of newly designed garment with standardized test scenes of typical motions.

In this thesis we address the physically-based simulation of cloth mainly in the context of the first of the above scenarios. We present the work realized throughout the BMBF[1]-funded research project *Virtual Try-On*. The cloth simulator *TüTex* developed by the computer graphics group at the University Tübingen (WSI/GRIS) was implemented during this project as part of a complete pipeline for the virtual try-on of cloth. The physical simulation of cloth is a very complex challenge with various facets. Hence, in our group several people worked on TüTex focussing on the physical model, numerical solver, collision detection and post-processing. The methods presented in this thesis focus on the two latter aspects (Figure 1.1). Though we mainly present applications in the field of cloth simulation, the proposed algorithms can also be employed for the simulation of other deformable objects like human tissue or hair.

1.2 Preliminaries

The simulation of cloth is a demanding task which requires the solution of various problems in the fields of computer science, mathematics, and physics. In the following, we give an overview of the intertwined steps of the simulation pipeline. The areas within the pipeline where this thesis contributes to are highlighted red in Figure 1.2.

The cloth itself as well as the remaining **geometric scene data** like avatars is represented geometrically as polygonal meshes. Due to their flexibility, commonly triangular meshes are used. To simulate the **mechanical behavior of cloth** internal and external forces are considered. Physical material properties mapping the internal forces within the cloth are therefore assigned to the meshes using different approaches. Until recently, discrete methods like mass-spring and particle systems were widely used to model these internal forces, namely tension, shearing, bending, and transversal contraction [31, 127, 48, 25]. Only within the last

[1] Bundesministerium für Bildung und Forschung

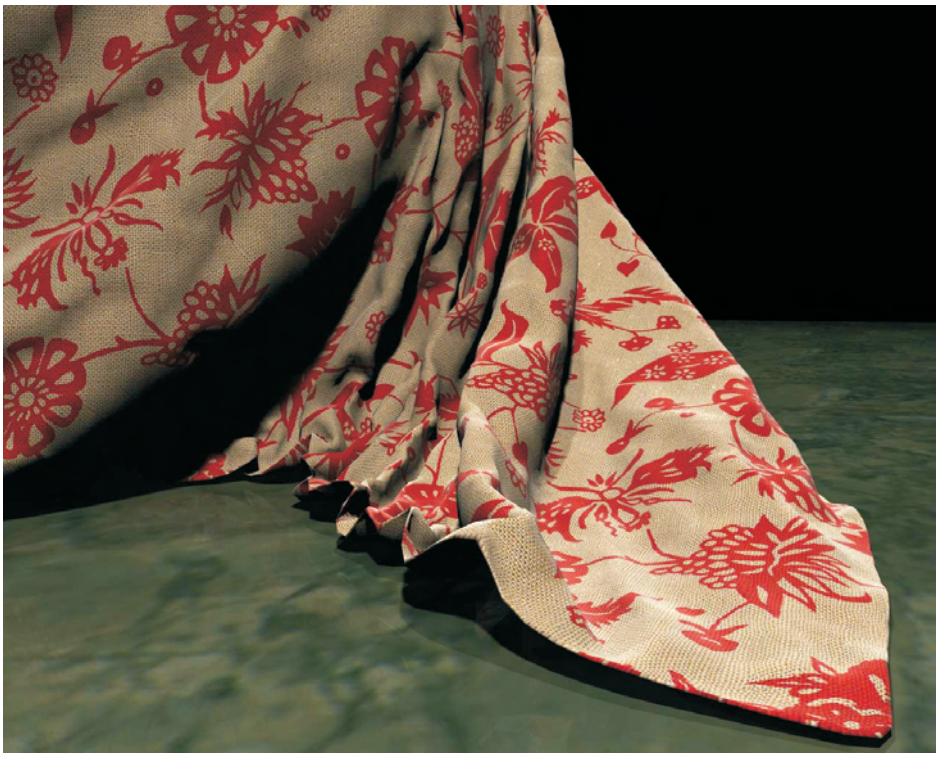


Figure 1.1: Detail of a simulation of a piece of cloth draped over a rotating sphere. This image shows the complex collision and self-collision situation that is detected and resolved using the algorithms presented in Chapter 2 and 3. The cloth is post-processed using the collision-free subdivision detailed in Chapter 4.

years, models based on continuum mechanics using finite elements have been employed that yield a similarly high performance and map material parameters more realistically [51]. Beside the internal forces, various external forces are taken into account. Especially gravity as well as air resistance and wind effects are modeled. Based on internal and external forces the equation of motion of cloth is set up using Newton's second law $F = ma$. To solve the emerging differential equation and to obtain the discrete simulation steps, a wide range of **numerical solvers** has been presented [64].

An essential task within the simulation process is the realization of interactions between different objects. The most important interactions are collisions between the scene objects. In nature collisions between solid objects are modeled and interpenetrations are avoided by very short ranging repulsive forces[2]. When calcu-

[2]This force that avoids interpenetration of objects follows from the occurring overlap of the electron clouds of the approaching atoms and the resulting quantum mechanical effects. In the

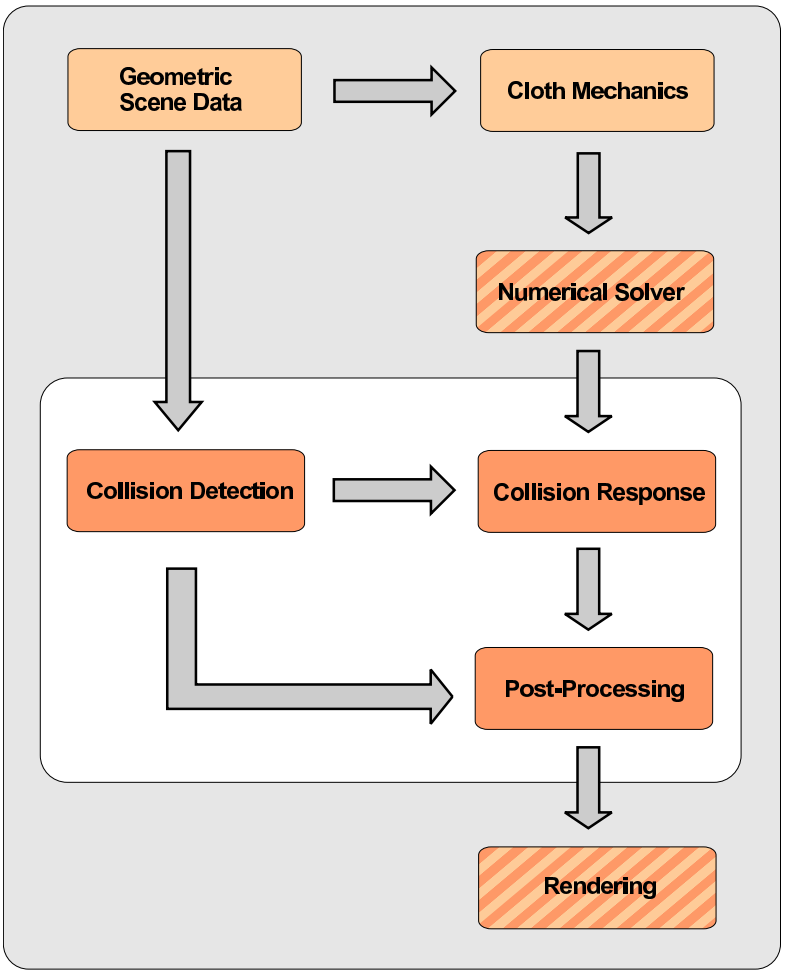


Figure 1.2: Overview of the cloth simulation process. The areas where this thesis contributes to, are shown in red.

lating the motion of objects with large time step sizes of up to some hundredths of a second, which are necessary to achieve fast simulations, these potential fields are easily tunneled. Hence, the repulsive forces are not included in the system and the simulated objects only move based on the remaining internal and external forces. This necessarily leads to interpenetrations after the numerical integration step has calculated new positions. Highly deformable objects may even intersect themselves resulting in self-collisions. In computer graphics strategies have evolved

hypothetical case that the atomic nuclei were coincident the electrons of two atoms would have to share the same orbital system. According to the Pauli Exclusion Principle no two electrons can share the same state so that in effect half the electrons of the system would have to go into orbits with an energy higher than the valence state. The resulting repulsive force F is modeled using a Lennard-Jones potential and is a function of the distance r of the two atoms: $F \sim r^{-12}$.

that solve these problems by detecting close features of the virtual objects and responding to these collisions or proximities (objects closer than some minimum distance to each other) by separating the objects. Numerous approaches have been proposed for this purpose in applications such as robotics, games, surgery simulation, and cloth simulation [30, 24, 127, 89]. As it is necessary to detect collisions between all involved primitives this **collision detection** step is a bottle neck in the simulation pipeline. While many of the original collision detection methods primarily address the problem of rigid body collisions, the approaches presented in this work focus on deformable objects. There are various aspects that complicate the collision problem for deformable objects like the higher number of potential collisions and the need for more complex **collision response** schemes. Collision response is the second step in collision handling, which resolves emerged collisions or proximities. These methods separate the colliding objects e.g. by altering their positions or velocities.

In order to obtain visually pleasing cloth simulations usually a high resolution of the underlying meshes is necessary. If coarse meshes were used, the polygonal structure would become visible or it might be impossible to model fine wrinkles or folds. However, though the performance of numerical solvers and collision detection methods has increased rapidly over the recent years, a high resolution of the underlying cloth meshes still requires too much computation time, especially for real time or interactive applications. Therefore it is often necessary to simulate clothing with only a few thousand triangles. To nevertheless obtain results with high quality, **post-processing** methods are used to either smooth the simulated meshes using a refinement or enhance them geometrically with folds and wrinkles. As this procedure does not influence the simulation itself, only the specific time steps of the simulation passed to the rendering may be post-processed. In the **rendering step**, the cloth is visualized together with its environment taking the specific surface properties and lighting conditions into account. Therefore material properties like refraction and reflection parameters can be applied to obtain a realistic look and feel of textiles.

1.3 Overview and Contributions

This thesis makes substantial contributions to all the fields highlighted in Figure 1.2. New approaches and methods are presented for collision detection, collision response, and post-processing of virtual garment. Additionally, the integration of all these methods into the simulation framework TüTex was realized and is described in this thesis. All contributions are detailed in the remainder of this thesis that is organized as follows. Chapter 2 covers collision detection methods used

in the simulation of virtual textiles. Foremost, we detail the current state of the art of the relevant collision detection approaches for deformable objects. These methods range from bounding volume hierarchies and distance fields to image-space techniques. We then give a comprehensive answer to the question, which of these collision detection methods best meets the demands of our specific problem of textile simulation. Therefore, we expose the specific advantages and disadvantages of these methods when applying them to highly deformable objects. We focus on the special requirements for dynamic scenes and self-collisions as well as the possibility to balance quality versus speed of the detection process. It turns out that bounding volume hierarchies fit best to detect collisions in our context. However, because they do not allow to balance speed versus quality, we introduce a new approach that uses a stochastic sampling of the colliding primitives to overcome this limitation in this chapter. Furthermore it yields a higher performance of both collision and self-collision detection. Finally, to detect collisions not only at a definite time, we present a continuous collision detection method. Using this method for deforming triangle meshes, it is possible to find the exact point in time at which the mesh primitives collide. Here we remark that later in this thesis continuous collision detection is primarily used to detect collisions introduced by post-processing methods.

Collision response schemes employed for textile simulation are discussed in Chapter 3. These methods are used to compute a collision-free state of the simulated cloth after collisions have occurred and were detected with the presented detection schemes. As any collision response scheme is only a limited model to reality, it is necessary to find an algorithm to alter positions and velocities of the mesh vertices or to constrain their motion in such a way that all collisions are resolved. First, we detail how such a geometric collision response has to be applied to a triangle mesh. Then, we present a collision response method that constrains the possible directions of motion in order to avoid further interpenetrations. As this method is not able to reliably resolve collisions for animated rigid objects, we detail a new method that alters positions and velocities while considering impulse conservation laws. This impulse-based collision response stably resolves all collisions and self-collisions in TüTex both for static and dynamic environment scenes.

As simulating cloth with high resolution meshes still results in long computation times and coarse meshes show an unsightly polygonal silhouette, we propose a post-processing method based on different subdivision schemes used to smooth coarse meshes (Chapter 4). Performing the physically-based simulation on coarse meshes followed by a subdivision step significantly accelerates the simulation process. In this regard, this thesis is the first to compare interpolating and approximating subdivision methods with respect to their results and practicability

in cloth simulation. Due to position alterations, the subdivision step may introduce intersections. To resolve them, we combine this step with the continuous collision detection and a collision response heuristic. The collision-free subdivision yields a substantial quality improvement without the necessity to simulate high resolution meshes.

While the collision-free subdivision only smoothes meshes, in Chapter 5 a heuristic is presented that enhances a coarse cloth representation with very realistic folds and fine wrinkles using textures. Our main contribution in this context is the automatic generation of the textures based on the deformation tensor of the physical simulation considering area conservation. While previous approaches depend on tedious user interaction, we present a new method that fully automatizes the generation of these textures. To obtain a high resolution cloth mesh we present an innovative method to combine these wrinkle textures with the collision-free subdivision described in the preceding chapter.

The application of the previously described methods is shown in Chapter 6. Here we detail the cloth simulation engine TüTex and show how the presented methods are implemented in this software framework. The application of the TüTex system as part of the automatic simulation pipeline of the research project *Virtual Try-On* is shown. Moreover, to facilitate the set up process of simulation scenes we developed the Maya plugin *tcCloth* based on TüTex. It combines the simulation engine with a powerful modeling environment to design and sew clothes and render the achieved results. To conclude this thesis, Chapter 7 summarizes the results of this work and gives an outlook where further research in the area of collision detection and post-processing of virtual textiles may be directed to.

CHAPTER 2

Collision Detection

2.1 Introduction

The detection of collisions is one of the most important tasks in the simulation of garment. Beside the physical simulation engine and numerical solver which compute the behavior of the cloth due to internal and external forces, a sophisticated collision detection scheme is necessary to yield convincing simulation results. Such a scheme detects collisions of the simulated cloth with its environment or with itself. Since cloth is extremely flexible and shows a very low bending resistance there are various aspects that complicate the collision detection problem compared to rigid bodies:

- In order to realistically simulate interactions between deformable objects, all contact points including those due to self-collisions have to be considered. For rigid body collision detection self-collisions usually do not occur.
- Efficient collision detection algorithms often use different spatial data structures including bounding volume hierarchies, distance fields, or alternative ways of spatial partitioning in order to accelerate the detection process.

These object representations are commonly built in a pre-processing step and perform very well for rigid objects, as the time spent in this step can be neglected and the spatial data structures can be kept for further simulation steps. For deforming objects, however, it is necessary to update or rebuild these data structures after each deformation step what primarily results in a reduced overall efficiency.

- Collision response for deformable objects is very complex and particularly for thin collision objects like cloth a complete through passing is possible. Hence, to yield a stable response the collision detection algorithms need to deliver detailed information such as penetration depth or direction of the collision objects.

The methods presented in this chapter are not restricted to collision detection between cloth objects. They can also be used to other applications such as surgery simulation, where collisions occur between deformable organs or between surgical tools and deformable tissue. In this case also topological changes due to cutting have to be taken into account.

After this introduction, the remainder of this chapter is organized as follows: As many different collision detection methods exist and have already been applied for cloth simulations, it is no simple task to select the method that matches best the needs of a specific problem or application. Hence, in Section 2.2 our contribution is to compare and classify the currently used collision detection methods for deformable objects with respect to applications in cloth simulation. Therefore, we present the state of the art of these approaches and identify their respective advantages and disadvantages to give the potential user a decision guidance for choosing the right method. Since most currently used collision detection methods do not allow to trade quality for speed, in Section 2.3 we detail on a new approach to stochastic collision detection. The presented stochastic method finds local proximities or interpenetrations by iteratively calculating the distance of sampled primitive pairs. We combine this technique with a bounding volume hierarchy to further accelerate the local distance minima search by restricting the primitive sampling to already close object parts. Subsequently, we demonstrate the high performance of this method and how it is possible to further speed up the detection process while reducing the detection quality. As collisions cannot only occur at the discrete time steps of a simulation but also in-between, continuous collision detection methods are employed. If such methods are not utilized, collisions may be missed or two objects may pass each other completely. In Section 2.4 we present such a continuous collision detection method that is used in a second collision detection phase, after close triangles have been detected by employing bounding volume hierarchies. This continuous collision detection scheme

is the basis for the collision-free post-processing methods presented in Chapter 4 and 5.

2.2 Detection Methods for Deformable Objects

In this section the prevalent, previously used methods for collision detection of deformable objects are analyzed. First, we present the state of the art and point out the specific characteristics of each approach. In Section 2.2.5 we then compare all detection methods by opposing the specific advantages and disadvantages especially in the context of cloth simulation. To decide which detection method suits best the specific needs of an application or problem, we summarize our results in a decision matrix.

This section is based on our survey articles [10, 11, 15]. Our work on bounding volume hierarchies has been published in [9, 8].

2.2.1 Bounding Volume Hierarchies

The idea of bounding volume hierarchies (BVHs) is to partition each object into a set of primitives enclosed by BVs. BVHs speed up the collision detection in two ways: First, a collision detection between BVs is much faster than a collision check between the objects enclosed by the BVs. Second, the hierarchy is used to efficiently prune unnecessary collision tests in areas where no collisions occur. BVHs have been applied for rigid body collision detection since the mid 1980s. For the collision detection of deformable objects they have proven to be a very efficient data structure as well.

In general, BVHs are defined as follows: Each node in the tree is associated with a subset of the primitives of the object, together with a BV that encloses this subset with a smallest containing instance of some specified class of shapes. A simple example for a triangle mesh with a hierarchy over spheres is illustrated in Figure 2.1. The bounding volumes that have been used for BVHs range from spheres [107, 73, 100], oriented bounding boxes (OBBs) [58], discrete oriented polytopes (DOPs) [81, 137], Boxtrees [138, 18], axis aligned bounding boxes (AABBs) [38, 124, 87], spherical shells [84, 83], to convex hulls [49]. Although this wealth of BVs has been proposed (Figure 2.2), two types deserve a special consideration: OBBs and k-DOPs. k-DOPs are convex polyhedra defined by k half-spaces. In order to turn the intersection test for the polyhedrons into simple interval tests, the hyper-planes have to form $k/2$ parallel pairs. A special case of k-DOPs with $k = 6$ are AABBs. k-DOPs are built and updated by enlarg-

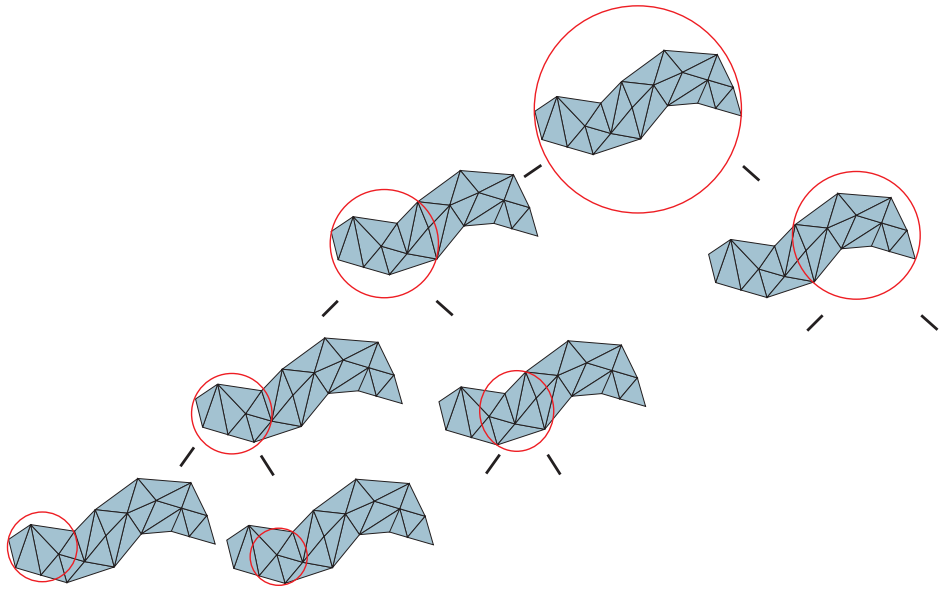


Figure 2.1: Four levels of a binary tree of bounding spheres over a triangle mesh.

ing all $k/2$ intervals, so that all points lie within the two half-spaces. OBBs are rotated AABBs, so that they are aligned with the enclosed object. To align the OBBs the convex hull of the object is calculated. The covariance matrix and the corresponding eigenvectors to this convex hull then define the orientation of the OBB. The update of OBBs is only efficient for translations or rotations of the complete enclosed object. For the intersection test projections on up to 15 axes are necessary [58]. OBBs have the nice property that, under certain assumptions, their tightness increases linearly as the number of polygons decreases. k-DOPs, on the other hand, can be made to approximate the convex hull arbitrarily exact by increasing k.

Using these BVs, in a pre-processing step before the actual detection step, for each of the possibly colliding objects a BVH is constructed. The object is partitioned until some leaf criterion in the hierarchy is met. Most often, each leaf contains a single primitive, but one could as well stop when a node contains less than a fixed number of primitives. Here, primitives are the entities which make up the graphical objects, which can be polygons, polyhedra, etc.

2.2.1.1 Hierarchy Traversal

For the collision test of two objects or the self-collision test of one object the BVHs are traversed top-down and pairs of tree nodes are recursively tested for

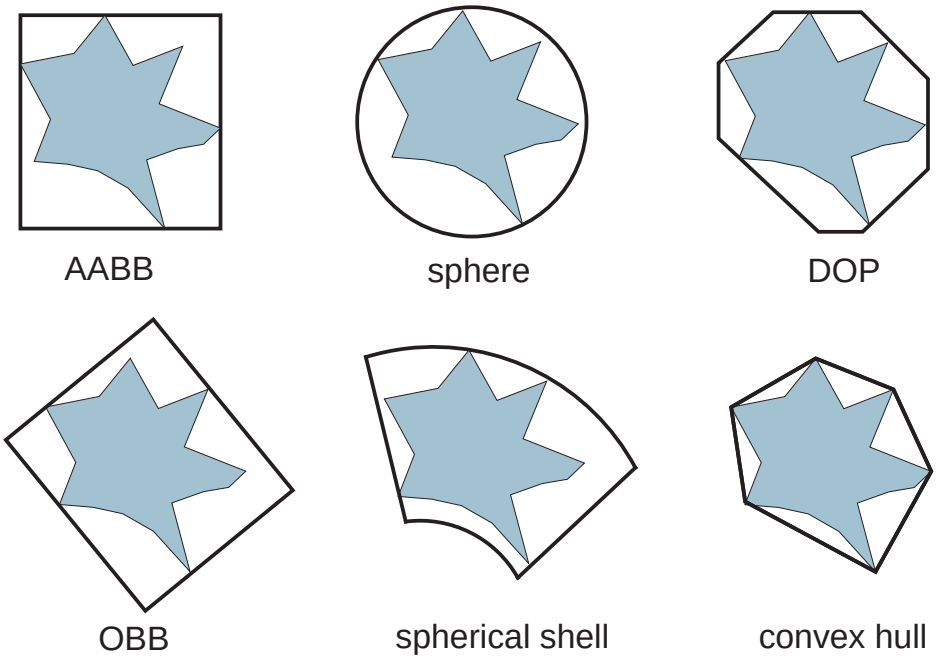


Figure 2.2: A variety of bounding volumes used for hierarchy-based collision detection.

overlap. If the overlapping nodes are leaves, then, in a further collision detection step, all enclosed primitives are tested for intersection. If one node is a leaf while the other one is an internal node, the leaf node is tested against each of the children of the internal node. If, however, both of the nodes are internal nodes, it is tried to minimize the probability of intersection as fast as possible. Therefore, van den Bergen [124] tests the node with the smaller enclosed volume against the children of the node with the larger enclosed volume.

For two given objects with the BVHs A and B, most collision detection algorithms implement the following general algorithm scheme:

```
traverse(A,B)
if A and B do not overlap then
   return
end if
if A and B are leaves then
   return intersection of primitives enclosed by A and B
else
   for all children A[i] and B[j] do
      traverse(A[i],B[j])
   end for
```

end if

This algorithm quickly zooms in on pairs of nearby polygons. Note that mixed cases where one node is a leaf and the other is an inner node are omitted. The characteristics of different hierarchical collision detection algorithms lie in the type of BV used, the overlap test for a pair of nodes, and the algorithm for the construction of the BV trees.

2.2.1.2 Construction of Bounding Volume Hierarchies

For rigid bodies, the goal is to construct BVHs such that all subsequent collision detection queries can be answered as fast as possible on average. Such BVHs are called "good" in the context of collision detection. With deformable objects, the main goal is to develop algorithms that can quickly update or refit the BVHs after a deformation has taken place. At the beginning of a simulation, a good BVH is constructed for the undeformed object just like for rigid bodies. Then, during the simulation, the structure of the tree is usually kept, and only the extents of the BVs are updated. Since for deformable objects, where not only rotations or translations of the enclosed objects occur, DOPs are generally faster to update and are preferred over OBBs. Since the construction of a good initial BVH is important for deformable collision detection, we will discuss some of the issues in the following, while efficient ways of updating the hierarchy are discussed in Section 2.2.1.3.

There are three different strategies to build BVHs, namely top-down [81, 8], bottom-up [111], and insertion [57]. However, the top-down strategy is most commonly used for collision detection and therefore here described in more detail. The idea of a top-down construction is to recursively split a set of object primitives until a threshold, e.g. a minimum number of enclosed primitives, is reached. The splitting is guided by a user-specified criterion or heuristic that yields good BVHs with respect to the chosen criterion. A very simple splitting heuristic was proposed by Gottschalk et al. [58]. First, each polygon is approximated by its center. Then, for a given set B of such points, its principal components (the eigenvectors of the covariance matrix) are computed and the largest of them, i.e. the one exhibiting the largest variance, is chosen. Subsequently, a plane orthogonal to that principal axis is placed through the barycenter of all points in B that splits B into two subsets. Alternatively, the splitting plane can be placed through the median of all points. This leads to a balanced tree. However, it is unclear, whether balanced trees provide improved efficiency of collision queries. Also the k-DOP hierarchy used in TüTex employs a top-down splitting to build the hierarchy. Each k-DOP is split according to its longest axis. The longest axis is determined by the

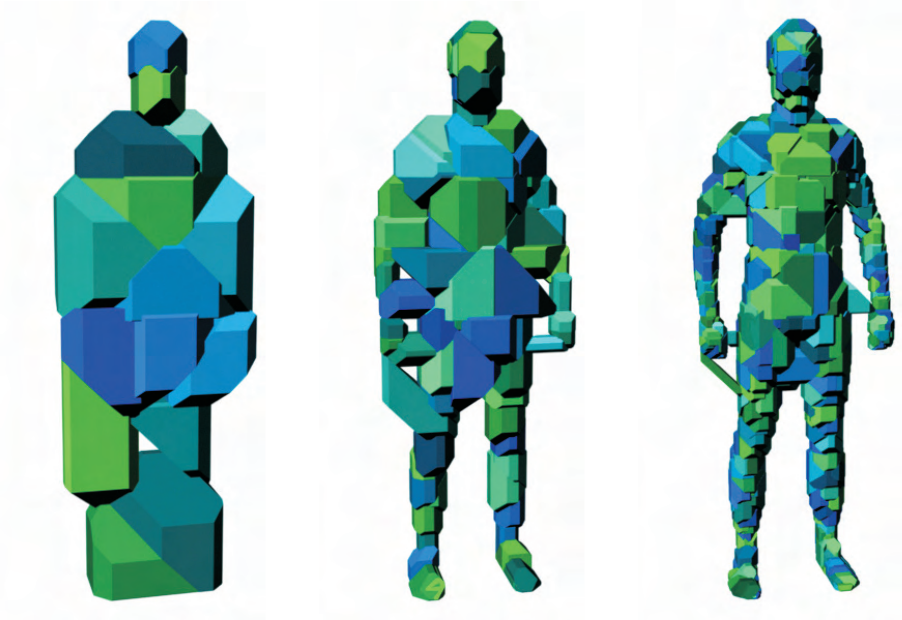


Figure 2.3: Three levels of an 18-DOP-hierarchy created top-down by splitting the parent DOPs along the longest axis.

triangle pair with the largest distance. Then the k-DOP is split parallel to this triangle pair through its center [9, 8]. Figure 2.3 shows three hierarchy levels of a corresponding 18-DOP-hierarchy over an avatar.

Volino and Magnenat-Thalmann [128, 129] as well as Provot [106] use a fairly different approach for deformable 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 and Magnenat-Thalmann use 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 clustering of faces 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 as described in Section 2.2.1.4.

Another crucial point is the arity of the BVH, i.e. the number of children per node. For rigid objects, binary trees are commonly chosen. In contrast thereto it has been shown that 4-ary trees or 8-ary trees give a higher performance for deformable objects [87, 8]. 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 not as high.

2.2.1.3 Hierarchy Update

In contrast to hierarchies for rigid objects, where the complete object undergoes a transformation, hierarchies for deformable objects need to be updated in each time step. Principally, there are two possibilities: refitting or rebuilding. Refitting adapts the BVs to the new positions of the enclosed primitives while rebuilding completely reconstructs the BVH. Refitting is much faster than rebuilding, but for large deformations, this leads to less tight fitting BVs and therefore to larger overlap volumes. For an AABB hierarchy of deformable objects, van den Bergen [124] has found that overall refitting is nevertheless faster compared to a complete rebuild.

Different strategies have been proposed not only for building a hierarchy but also for the hierarchy update by refitting. Larsson and Akenine-Möller [87] 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 first only updates the top half of the hierarchy bottom-up. If, during the hierarchy traversal, 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 requirements because they have to store the leaf information about vertices or faces also in the internal nodes.

2.2.1.4 Self-Collision Detection with BVHs

BVHs can be easily employed to accelerate self-collisions, which is particularly important for deformable objects. In general, collisions and self-collisions are performed by the same algorithm using BVHs. If several objects are tested for collisions, the respective BVHs are checked against each other. Analogously, self-collisions of an object are detected by testing one BVH against itself.

However, it has to be noted that BVs of neighboring regions can overlap, even though there are no self-collisions. To eliminate such cases efficiently, different heuristics have been presented. Volino and Magnenat-Thalmann [128] proposed an exact method to avoid unnecessary self-intersection tests between certain BVs. The idea is based on the fact that connected regions with sufficiently low curvature cannot self-intersect, assuming that their contour is convex. Therefore, a vector

with positive dot product with all face normals of the region is searched. 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. A very similar approach is used by Provot [106] who uses normal cones representing a superset of the normal directions, which are calculated for each convex region. The cone's apex angle α represents the curvature, indicating possible intersections if $\alpha \geq \pi$.

2.2.2 Distance Fields

Another collision detection method widely used to find intersection between cloth and rigid objects are distance fields. Originally distance fields were not employed for collision detection but were used in computer graphics for volume rendering [56] and to create offset surfaces [101]. A distance field is a scalar field that specifies the minimum distance to a surface. If the surface is closed, the distance may be signed in order to distinguish between the inside and outside of the shape. The collision detection between points and the object is performed by evaluating the signed distance. Representing a closed surface by a distance field has the advantage that there are no restrictions on the topology. Additionally, the described evaluation of distances and the evaluation of normals needed for the collision detection and response process is very fast and independent of the complexity of the object.

Mathematically, a distance field $D : \mathbb{R}^3 \rightarrow \mathbb{R}$ defines a surface S as a zero level set, i.e. $S = \{p|D(p) = 0\}$. Contrary to other implicit representations, a distance field evaluation not only yields information about interior and exterior, but also the distance to the closed surface. The distance can be specified using different metrics, for collision detection usually the Euclidian distance metric is used. The set-up of the distance field is usually done in a pre-processing step as the collection of this information may be very time consuming. In the following we present the different object representations used for distance fields and the corresponding techniques for the distance field computation. Then we address the collision detection process between deformable and rigid objects using distance fields.

2.2.2.1 Distance Field Representations

To represent distance fields different data structures for space partitioning like uniform 3D grids, octrees and binary space partitioning (BSP) trees have been proposed. The simplest distance field representation are uniform grids, where the distance values are computed for each grid point. In-between values can be com-

puted by trilinear interpolation. The major advantages of these uniform grids are the simplicity to implement them and the constant time necessary for distance queries at any of the grid points. The normals of the surface necessary for most collision response methods can easily be computed from the analytic gradient of the trilinear interpolation. Beside these advantages, however, uniform grids have drawbacks, resulting from their constant sampling rate. As fine details, e.g. sharp features, require dense sampling, immense volumes are needed to accurately represent the distance fields with uniform grids, even when these fine details occupy only a small fraction of the volume.

A possibility to overcome these problems are the adaptively sampled distance fields (ADFs) presented by Frisken et al. [53], where a "detail-directed" sampling is used. High sampling rates are used in regions where the distance field contains fine detail and low sampling rates where the field varies smoothly. This sampling strategy permits arbitrary accuracy in the reconstructed field together with efficient memory usage. The distance data of ADFs is organized in a hierarchy for fast localization. Although various spatial data structures are suitable for this purpose in general, ADFs are usually stored at cell vertices of an octree. The construction of the ADF uses the following strategy. Each cell of the octree is further subdivided as long as the result of the used reconstruction scheme, i.e. the trilinear interpolation, does not properly approximate the original distance field. Contrary to a standard 3-color octrees where each cell, which is not completely inside or outside, is subdivided, for ADFs the subdivision stops when a maximum tree depth is reached. The major advantage of ADFs is their good compression ratio, however, for collision detection purposes, special care has to be taken in order to guarantee continuity between different levels of the tree. Whenever a cell is adjacent to a coarser cell, its corner values have to be altered to match those of the interpolated values at the coarser cell [34, 134].

Using BSP-trees as distance field representation, the memory consumption can be reduced even further. Wu and Kobbelt [135] achieve this using a piecewise linear approximation of the distance field, which is not necessarily continuous. In their work, also several algorithms for selecting appropriate splitting planes of the tree are proposed. The drawback of the BSP-tree representation are the high costs for the construction and the problems arising from discontinuities between cells, since they cannot be resolved as easily as for ADFs.

2.2.2.2 Distance Field Computation

After showing different representations for distance fields, we discuss its actual computation. In the applications considered in this thesis, the surface of the collid-

ing objects is given as deforming triangular meshes. In this context it is sufficient to compute the distance values only in a small band near the surface that can be traversed by the colliding objects within one time step. For this computation step there are three different approaches: methods based on Voronoi diagrams, propagation methods, and methods using trees. A tree data structure can be used to calculate the distance of each grid point independently. Remote triangles, not influencing the distance calculation can easily be culled. For this process, Payne and Toga used BVHs [101]. Also octrees have been applied for this step, but all these approaches have shown not to be competitive compared to other methods. Propagation methods start with a narrow band of distances computed near the triangular surface. This contour is then propagated over the whole volume to compute the distance transformation from the neighbors. The level set method presented by Sethian [113] and distance transforms [74] are two examples of propagation methods.

Hoff et al. [69] accelerate the computation of distance fields using graphics hardware. They render a polygonal approximation of the distance function on the depth-buffer hardware and compute the generalized 2D and 3D Voronoi diagrams. This approach builds distance meshes for each Voronoi site and works on any polygonized geometric model. A scan-conversion algorithm to compute the Euclidean distance in a narrow band around triangle meshes has been proposed by Mauch et al. [32, 92]. They use the connectivity of the mesh to compute polyhedral bounding volumes for the Voronoi cells. The polyhedrons are cut into slices along grid rows and the resulting polygons are scan-converted in order to determine which grid points lie inside. The distance for inner grid points is easily computed as the distance to the Voronoi site. Sigg et al. [116] improved this algorithm by evaluating fragment programs of the graphics hardware to scan-convert the polyhedrons. They also reduced the number of polyhedra which have to be scan-converted. The drawback of algorithms based on graphics hardware is that collision queries usually require reading back the distance field to the CPU. These readbacks are still very slow on current graphics boards [118]. For triangular meshes without any adjacency information, Fuhrmann et al. [54] proposed an algorithm which computes distance values independently for each triangle without using Voronoi cells. For finely tesselated triangle meshes, they have shown that except for some sign errors, this technique is able to efficiently compute distance fields.

2.2.2.3 Collision Detection with Distance Fields

When using distance fields, the collision detection process between different objects is carried out pointwise. Vertices of one object are checked against the dis-

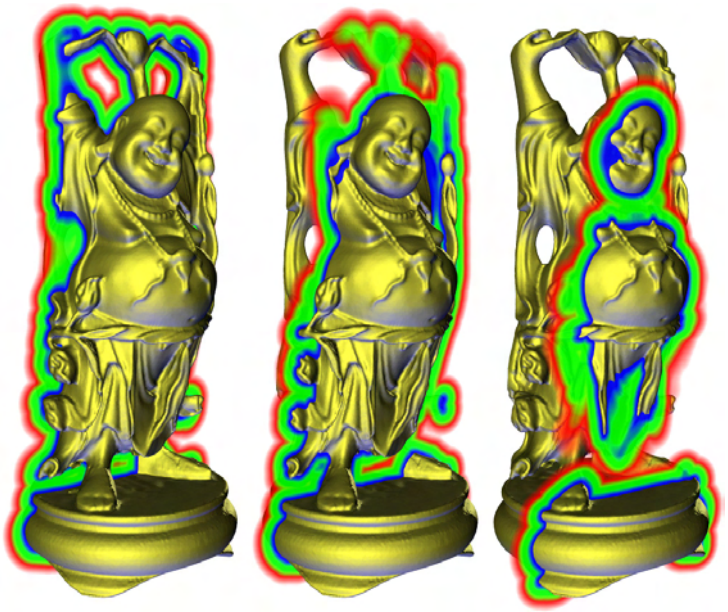


Figure 2.4: Three slices through the color-coded distance field of the Happy Buddha. The distance field is only evaluated within a close band near the surface. Image courtesy of A. Fuhrmann, IGD Darmstadt.

tance field of the other object and vice versa. A collision has occurred if $D(p) < 0$. In the case of cloth simulation, only vertices of the cloth mesh are tested for collisions. As collisions are not only caused by vertices interpenetrating the other object, but also by colliding edges it is necessary to offset the vertices from the zero isosurface by a predefined distance ε (Figure 2.5). This offset depends on the mesh resolution. Fuhrmann et al. [54] additionally apply a subdivision-like approach by additionally testing the center of each edge in the deforming mesh in order to improve the precision of collision detection (Figure 2.6).

As for BVHs, when applying distance fields to deformable bodies, an update of the distance field representation is necessary after each deformation, i.e. after each time step. The update of the distance field is a common bottleneck of all methods described above. For volumetric bodies with only small deformations, Fisher and Lin [52] proposed an efficient update algorithm that is, however, not intended for real-time use. They employ an internal distance field created by a fast marching level set method to propagate distance information only inside the objects. The actual collision detection is carried out by a hierarchical method. During collision response the distance fields are deformed due to the geometry and used for an approximation of the penetration depth. This method is able to

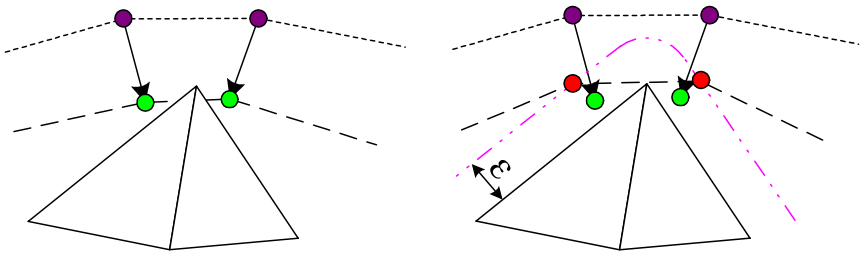


Figure 2.5: When resolving collisions based on distance fields, an offset is necessary, as only vertices are checked against the distance field while collisions between edges are neglected. Without offsetting the vertices inter-penetration artifacts may occur during collision detection (a). Introducing an ε-offset can solve this problem (b). Image courtesy of A. Fuhrmann, IGD Darmstadt.

handle both collisions and self-collisions. However, their approach is only well-suited for volumetric objects. Thin objects like cloth cannot be represented by internal distance fields and therefore no self-collision detection is possible. Bridson et al. [34] employ ADFs for detecting collisions between clothing and animated characters. These animated characters deform over time based on the underlying skin and muscle simulation. The corresponding distance fields for each time step are pre-calculated using a fast marching method. For computing distance fields and to accelerate distance field updates, Vassilev et al. [125] proposed an image-space approach. They use rendering hardware for the construction of two depth and normal maps of the object, one map for the back of the object and one map for its front. These maps are used for distance calculations and collision response. However, this algorithm is very inexact and is restricted to convex shapes or appropriate mapping directions in the case of animated characters. The approach is a mixture between distance fields and image-space techniques and is therefore further addressed in Section 2.2.4.

2.2.3 Spatial Subdivision

The idea behind spatial subdivision is that only the primitives of the object are tested for collision which belong to the same cell of a space partitioning. Hence, spatial subdivision algorithms usually proceed in several steps. In a first step, the primitives of the objects are assigned to the cells of a space partitioning. Then, in the second step, it is checked which primitives of the different collision objects are in the same cell. If these primitives are intersecting, a collision is detected. If

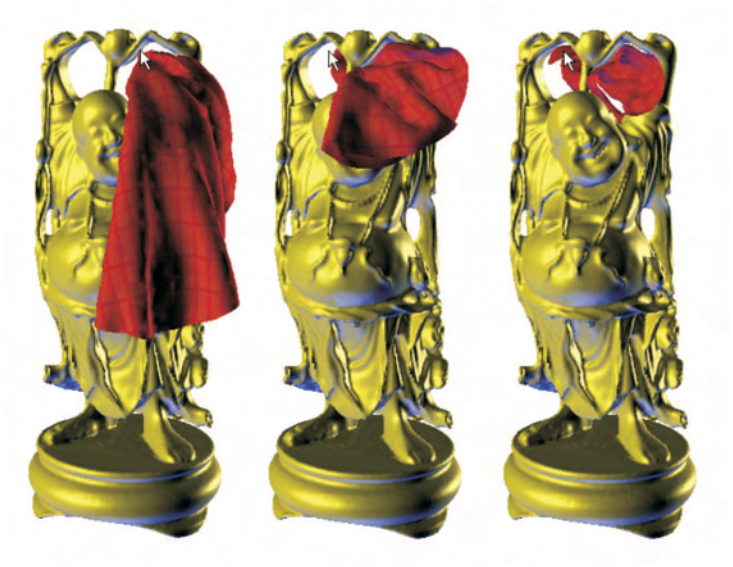


Figure 2.6: Interactive animation of cloth using the distance field of the Happy Buddha (Figure 2.4 for collision detection). Image courtesy of A. Fuhrmann, IGD Darmstadt.

both colliding primitives belong to the same object, a self collision is found. Like the previous presented collision detection methods, spatial subdivision also needs an update step if the objects move or deform. The assignment of the primitives to the grid cells has to be renewed in each time step.

Already in the 1960s and 1970s [88, 108] spatial subdivision has been proposed for neighborhood queries. Later, for the collision detection process of rigid objects various approaches have been presented that employ different representations for the space partitioning. Similar to the space partitioning used for distance fields uniform grids [122, 55, 139], octrees [23] or BSP-trees [93] are employed. Turk [122] was the first using spatial hashing for collision detection. Mirtich [95] used a hierarchical spatial hashing approach as part of a robot motion planning algorithm, which, however, is restricted to rigid bodies. Contrary to BSP-trees and octrees that are object-dependent, spatial subdivisions using uniform grids are independent of the objects and are therefore well suited for objects that deform with respect to the simulation time.

An approach combining spatial hashing and uniform grids for the detection of collisions and self-collisions of deformable tetrahedral meshes has been employed by Teschner et al. [120] (Figure 2.7). Their algorithm implicitly subdivides $\mathbb{R}^3$ into small grid cells and utilizes a hash function to map these 3D grid cells to a

hash table. This approach is memory efficient and additionally provides flexibility, since it allows to handle potentially infinite regular spatial grids. A further advantage is that no complex data structures like octrees or BSPs are required.

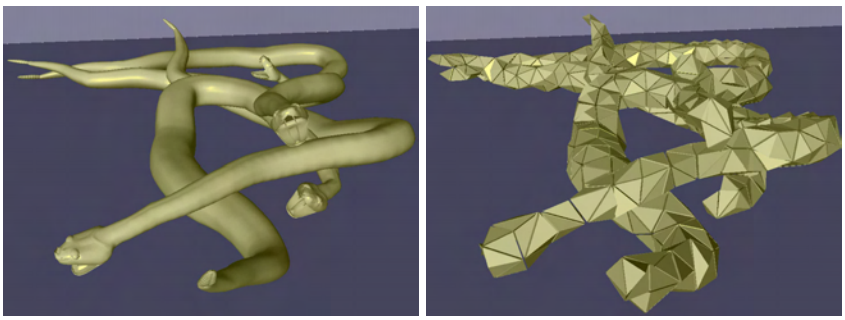


Figure 2.7: Interactive environment with dynamically deforming objects. For collision detection, a spatial hashing algorithm is used. Both, the surface with high geometric complexity and the underlying tetrahedral mesh are shown. Image courtesy of M. Teschner, University of Freiburg.

2.2.4 Image-Space Techniques

Another class of collision detection methods are image-space techniques. These algorithms render the objects by projecting them in various directions and then use the depth maps of the images to find intersections. To accelerate the detection process, these projections can be obtained using graphics hardware. Since image-space techniques do not require any pre-processing, they are especially appropriate for environments with dynamically deforming objects. In recent years a wide variety of publications have been presented in this field of research [99, 22, 20, 79, 65, 82, 59, 21].

One of the earliest approaches to image-space collision detection of convex objects has been presented by Shinya and Forgue [115]. They render the two depth layers of convex objects (front and back layer) into two depth buffers. The interval between the smaller and the larger depth value at each pixel then approximately represents the object and is efficiently used for interference checking. A similar approach has been presented by Baciu et al. [22], however, both methods are restricted to convex objects and do not take self-collisions into account. Myszkowski et al. [99] presented an image-space technique that lifted the restriction to convex objects and can detect collisions for arbitrarily-shaped objects. However, the maximum depth complexity remains limited and self-collision detection is not supported. Additionally, in a pre-processing step object primitives

need to be sorted. Therefore, this method cannot efficiently work with deforming objects, where the shape changes in each time step.

The first application of image-space collision detection to dynamic cloth simulation was published by Vassilev et al. [125]. As detailed in Section 2.2.2, they render an avatar from a front and a back view to generate a depth map. Using this depth map they can detect intersecting cloth vertices. Their approach, however, is restricted to convex objects what cannot be assured for deforming animated avatars. Hence, the simulation can show artifacts due to undetected collisions. In medical applications image-space techniques were employed by Lombardo et al. [89]. They detect intersections of a surgical tool with the simulated deformable tissue by rendering the interior of the tool.

Only recently, Baciu and Wong [21] presented a new image-space approach that extends the approach of Vassilev et al. to non-convex objects. They decompose the object's surface depending on the curvature and render these patches into the frame buffer. Then, however, they do not directly perform the collision detection process based on this information, but collect the potentially colliding triangle pairs for further processing. So this approach is a combination between image-space techniques and spatial subdivision. This configuration allows to detect both collisions and self-collisions. The algorithm of Kim et al. [79] combines BVHs with a multipass rendering approach to perform closest-point queries. Another approach able to detect collisions between arbitrarily-shaped objects is presented by Heidelberger et al. [65]. Similarly to Shinya and Forgue [115] they compute the layered depth image [114] for an object to approximately represent its volume. Heidelberger et al. are, however, not restricted to convex objects but only need a closed surface in order to have a defined object volume. In [66] they extend their work by combining the image-space object representation with information on face orientation to allow self-collision detection.

2.2.5 Comparison and Discussion

After the presentation of the state of the art for collisions detection between deformable objects, in this section we compare the advantages and disadvantages of the various methods and detail which choices influence the performance of the collision detection process. Furthermore, we provide the potential user with a decision guidance for choosing the detection algorithm out of this variety that matches best the specific needs of an application. For the application in the cloth simulator TüTex we identify the requirements and show which method should be used to detect collision and self-collision particularly for cloth simulations with animated avatars.

2.2.5.1 Bounding Volume Hierarchies

Regarding BVH methods, one of the most important choice that strongly influences the efficiency of the collision process is the used BV. In contrast to rigid body collision detection, the collision detection of deforming objects with BVHs requires frequent updates of the hierarchy. It has been shown that for these cases DOPs, and especially AABBs, should be preferred to other BVs, such as OBBs, because AABBs can be updated or refit more efficiently. OBBs approximate the objects tighter than AABBs but require more time to rebuild or update. Additionally, 4-ary or 8-ary trees have shown a better overall performance compared to binary trees. For applications where frequent topology changes occur, it is necessary to carefully choose the method to build and update the hierarchy.

Although deformable objects require frequent updates of BVHs, BVHs are nevertheless well-suited for animations or interactive applications, since updating or refitting of these hierarchies can be done very efficiently. Furthermore, BVHs can be employed to detect self-collisions while applying additional heuristics to accelerate this process. BVHs also work with triangles and tetrahedrons as object primitives, which allows for a more sophisticated collision response compared to a pure vertex-based response. For cloth simulation, the hierarchy can be built upon the cloth objects, taking neighborhood and curvature information into account. A major advantage of BVHs is their independence of a underlying grid or image resolution. Therefore they easily adapt to all collision configurations and do not miss any collision. Also BVHs can easily be adopted to allow continuous collision detection. A trade-off between quality and detection performance is, however, not possible.

2.2.5.2 Distance Fields

Distance fields can be employed to detect collisions in real time and even to detect self-collisions in non-interactive applications. The bottleneck of all distance field methods is the time needed for their generation and update. Recently, faster algorithms have been proposed but they are still not fast enough for real-time applications. Especially for deformable objects this poses a major problem. Therefore, distance fields are also not well suited for self-collision detection. For rigid, closed objects, however, distance fields provide efficient and robust collision detection, since they divide the space around them into inside and outside.

When only collisions between a piece of cloth and a static rigid object have to be detected, distance fields are very effective as the distance fields can be pre-computed. Distance fields efficiently deliver penetration depth and normals

needed for collision response. In order to decrease memory requirements or generation time, it is possible to reduce the resolution of the distance field, which results in lower accuracy. Hence, a balance between performance and accuracy is possible. Distance fields also have no problems with topology changes of the deformable object, as the collision detection is carried out pointwise. On the other hand there exists no effective strategy for continuous collision detection with distance fields.

2.2.5.3 Spatial Subdivision

The major advantage of spatial subdivision is its simplicity. The principal design choice is the used data structure to represent and subdivide the 3D space. This data structure strongly influences the efficiency of pre-processing and collision detection. Furthermore, the memory requirements have to be considered as they vary significantly between the various data structures and with the chosen subdivision level. Spatial subdivision can be employed to detect both collisions and self-collisions. Collision detection algorithms using spatial subdivision are independent of topology changes and are well suited for deformable or moving objects. The underlying primitives are not restricted to triangles if an appropriate intersection test is implemented for them. However, by these methods no balance between speed and detection quality is possible.

2.2.5.4 Image-Space techniques

As image-space techniques do not require time-consuming pre-processing they are especially well suited for dynamically deforming objects. They can be employed both for the detection of collisions and self-collisions. Furthermore, topology changes pose no problems. While image-space techniques usually are based on triangulated surfaces, they can also be extended to other object primitives as long as these primitives can be rendered.

Contrary to the other methods, image-space techniques rasterize the objects to detect collisions. Hence, they cannot provide exact collision information as long as they are not combined with one of the other detection methods. Additionally, the given information used for collision response is limited or needs further post-processing. As the accuracy of image-space techniques depends on the resolution of the underlying dicretization during the rendering process, performance and accuracy may be balanced. Using graphics hardware the rendering process and therefore the collision detection can further be accelerated. However, as buffer read-backs are still very slow and the flexibility of programmable graph-

Collision detection meth.	BVHs	Dist. fields	Sp. subd.	Image-sp. techn.
Topology changes	+/-	+	+	+
Balance speed/quality	-	+	-	+/-
Self-collision detection	+	-	+	+
Continuous coll. det.	+	-	+/-	+/-
Animated avatars	+/-	-	+/-	+
Coll. response inform.	+	+/-	+/-	-

Table 2.1: Comparison between different collision detection methods

ics hardware is limited, there are applications, where software implementation are faster than solutions in hardware.

2.2.5.5 Conclusion for TüTex

After this overview of advantages and disadvantages of the various presented collision detection methods, which is summarized in Table 2.1, we discuss which method should be used for TüTex. The application in TüTex requires the following properties:

- detect collisions and self-collisions
- work with animated avatars
- deliver collision response information
- compatible with continuous collision detection

Comparing these requirements with Table 2.1 we see that especially a bounding volume based approach is well suited for a cloth simulation system like TüTex. Additionally, as cloth can both span over a huge volume as well as being folded together to a very narrow area, for the collision detection process it is necessary to be independent of the resolution of underlying grids or depth buffer resolutions. A drawback of BVHs, however, is their lack of balancing performance and accuracy that is necessary in interactive applications. To deliver a BVH method that allows this balancing, we developed the hierarchy accelerated stochastic collision detection presented in the following section.

2.3 Hierarchy Accelerated Stochastic Collision Detection

Additionally to the collision detection methods presented in Section 2.2, recently a new class of collision detection approaches have emerged, so called "inexact" methods. Using these approaches it is possible to balance performance and detection quality. This is motivated by several observations. First, polygonal models are always just an approximation of the true geometry and the perceived quality of most interactive 3D applications does not depend on exact simulation, but rather on real time response to collisions [123]. At the same time, humans cannot distinguish between physically-correct and physically-plausible behavior of objects [27]. This leads to the conclusion that a degraded precision can be tolerated in order to improve the performance of the collision detection step.

In literature, currently two different stochastic approaches can be found. The first method proposed by Klein and Zachmann [80] uses probabilistic methods to estimate the possibility of collision with respect to a given quality criterion. With this method the quality of the collision detection can be specified by the user directly and ensures more control over the detection performance. This approach can be used to accelerate collision detection with BVHs but is restricted to rigid objects. The second method presented in the work of Guy and Debunne [61] as well as in Raghupathi et al. [109] initially "guesses" colliding pairs by a stochastic sampling within the colliding bodies. The exact colliding regions are then narrowed down by iteratively converging the sample pairs to local distance minima. As in most cloth simulation systems within two time steps of some hundredth of a second the objects only move a very small distance and also the relative position between the colliding object primitives mostly remains the same, also the spatial and temporal coherence is exploited. For rigid objects the use of temporal coherence between consecutive time steps has also been analyzed [38, 60].

As we want to develop a collision detection for deformable objects being applicable both for collisions and self-collisions, we build on a stochastic approach similar to the method presented by Raghupathi et al. The major disadvantage of this stochastic sampling is the generation of sample pairs on the complete objects. As many of these sample pairs are generated in areas laying far away from each other, most of them do not converge to a local distance minima relevant for the collision detection. To overcome this limitation, in the following we propose a new stochastical collision detection framework combining two detection algorithms. The first algorithm is an extension of the above shortly described stochastic collision detection method which will be detailed in Section 2.3.1. The second algorithm is based on a hierarchy of bounding volumes which helps us to

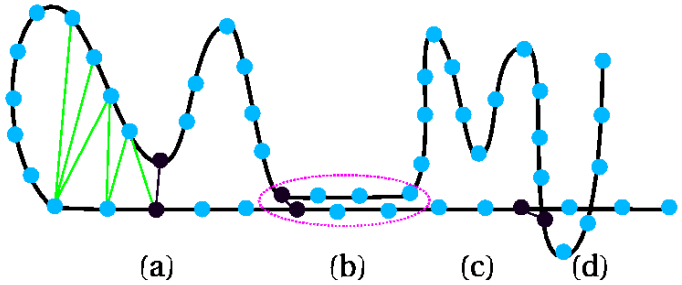


Figure 2.8: Self-collisions of a deformable object. The active pairs of points are shown in black; for clarity only vertex-vertex pairs are shown. An active pair has converged iteratively to a local distance minimum (a). A collision has been detected and the associated collision cluster is encircled (b). No active pair is present yet (c). A collision has been detected too late and an intersection already occurred (d).

pick appropriate random pairs. An overview of the design choices for this bounding volume hierarchy and the combination of the two algorithms to the *Hierarchy Accelerated Stochastic Collision Detection* is detailed in Section 2.3.2. Finally the obtained results and the validity of our approach is discussed.

The work presented in this section has been published in [5].

2.3.1 Active Pairs

The idea of the stochastic collision detection algorithm is to efficiently detect collision and self-collisions by tracking pairs of primitives (vertices, edges, triangles, etc.). These pairs of primitives are called *active pairs*. By choosing the number of newly generated active pairs, it is furthermore possible to trade detection completeness for computation speed. Using the active pairs method, the collision detection proceeds as follows: At first a set of active pairs is chosen randomly (Figure 2.8). Each of these pairs then iteratively converges to a local distance minimum by repeatedly replacing each element by its topological neighbor if their distance is smaller than the previous minimum (Figure 2.8(a)). If the distance falls below a given proximity threshold ε, a collision is detected. Particularly for cloth simulation, where large areas of the garment are in contact with the body, collisions are not isolated but are grouped in collision clusters (Figure 2.8(b)). If one initial collision within a cluster has been detected, the active pair method rapidly finds all nearby collision pairs during the described convergence process. For the

case that such an initial active pair is not present in a close area, as in Figure 2.8(c), this might eventually lead to an undetected collision (Figure 2.8(d)).

At each step of the animation loop, we update each active pair by iterating it until convergence is reached and the new local minimum is found. Due to temporal coherence in the animation, the number of necessary iterations is generally small. At each iteration of one pair, the neighboring pairs are visited for possible consecutive collisions, and detected collisions are added to a list. The possibly new closest pair replaces the current active pair. We actually store the visited pairs in a hash table and look for the closest not already visited neighboring pair to avoid tracking the same local minimum twice. Collision clusters are progressively detected either by growth around the currently detected collisions, or by the convergence of new pairs. If after the convergence the elements of an active pair are far from each other it is expected that the probability of detecting a collision by this active pair in the near future is very low. The active pair is thus considered useless and is discarded. The number of new active pairs per time step as well as the criteria used to discard useless pairs is tunable.

The active pair algorithm can be summarized as follows:

Active Pairs method
add colliding primitives of previous step to list of active pairs
generate new sample pairs and add to list of active pairs
for all active pairs **do**
  search in neighborhood for local minimum
  while local minimum not reached **do**
    calculate distance and add pair to hash table
    if distance $< \epsilon$ **then**
      add pair to list of colliding primitives
    end if
  end while
  if distance at local minimum $> \epsilon$ **then**
    remove pair from active pairs list
  end if
end for

By design, this method does not guarantee to find all the collisions unless all possible pairs are tried. We denote the *detection ratio* the number of detected collisions divided by the number of actual collisions. This ratio as well as the computation time devoted to collision detection depends on the number of active pairs we use. By tuning the number of active pairs the trade-off between detection ratio (quality) and computation time (speed) is possible.

In order to further accelerate the detection process, we start the converging

process by using pairs of points representing the primitives, since this results in faster distance computations. When the points are closer than twice the average primitive size, we switch to the calculation of the exact distances of the two primitives as this yields a higher accuracy and delivers more relevant information for the collision response process.

2.3.2 Combination with k-DOP Hierarchy

Using only a stochastic method to find new close regions between two meshes has a major drawback. The candidate pairs, where the iterative search for the close regions begins, are, by the design of the method, randomly distributed over the complete mesh. Nevertheless, often the close regions between two meshes are restricted to a rather small area of the complete mesh. Hence, most random sample pairs may be generated in areas without any collision.

To overcome this drawback of the pure stochastic collisions detection we combine it with a bounding volume hierarchy of k-DOPs. After presenting an overview of BVHs in Section 2.2.1 we now detail the design choices for the used BVH. We have chosen k-DOPs as bounding volumes, because they usually fit more tightly around meshes than spheres, while being much faster to update than OBBs. In order to detect all k-DOP proximities within a distance ε, we enlarge each k-DOP by an offset of $\varepsilon/2$ in each of its k directions. We efficiently built the hierarchy over these k-DOPs by using a top-down splitting method. In our implementation, first the longest side of a k-DOP is determined by the face pair with the maximum distance. Subsequently, the k-DOP is split parallel to this face pair through its center. As some polygons are generally cut by the splitting plane, they are assigned to that child node which would contain the smaller 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 turned out to be efficient on the one hand and to produce well balanced trees on the other hand. Previous approaches employed binary trees to store the hierarchy since they require the smallest number of overlap tests. However for the collision detection between deformable objects we have chosen 4-ary trees or 8-ary trees as they show a much better performance [8].

Generally, the hierarchy update re-inserts the vertices into the leaf k-DOPs and builds the inner k-DOPs by unifying the $k/2$ intervals of the child bounding volumes. Because this hierarchy update can be costly we developed a lazy hierarchy update. As described above, the bounding volumes are inflated by a certain distance $\varepsilon/2$. Then the hierarchy update is not needed as long as the enclosed

primitives did not move farther than the distance ε. Thus, the hierarchy update is accelerated for slow parts of the scene and for small time step sizes. As detailed in the following subsection this lazy update perfectly fits to our stochastic approach. A more detailed description of this k-DOP hierarchy also employed in TüTex can be found in [8].

Bounding volume hierarchies have been shown to be very efficient for collision detection of deformable objects. Nevertheless they have two major drawbacks. First, even with the lazy hierarchy update, the hierarchy needs to be updated frequently due to the deformations of the objects, which is very time consuming. Second, bounding volume hierarchies are usually built around triangles or tetrahedrons, and return colliding or nearby pairs of these, although an efficient and physically-based collision response is based on vertices. Therefore, in a second collision detection step, all close triangle pairs are checked for proximities between two edges, or a vertex and a triangle, where the necessary collision response can be easily applied.

Our new approach now combines the active pairs with the presented k-DOP hierarchy. In this combination, the drawbacks of both methods can be minimized. The process involves two steps: first a collision detection on the k-DOP hierarchy is performed, returning all colliding leaves of the hierarchy tree. Each leaf can contain a specified number of triangles or other primitives. Then in the second step the stochastic collision detection is performed, where the random sample pairs are created only on the colliding leaves (Figure 2.9).

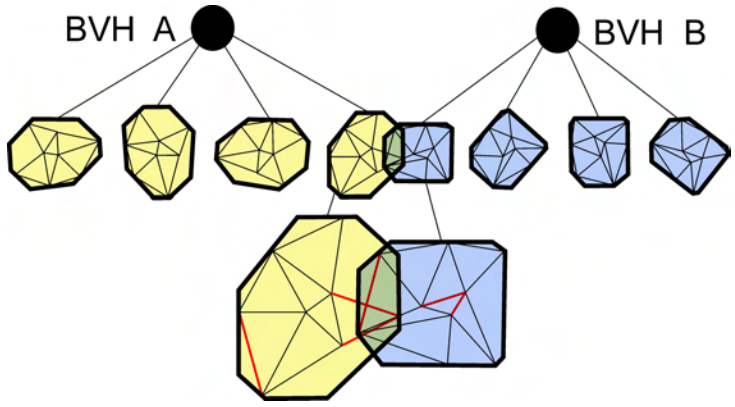


Figure 2.9: The creation of random samples is restricted to colliding leaves of the BVH.

The algorithm for the collision detection process between two objects with the corresponding BVHs A and B is then:

update(A and B) buttom-up as detailed in section 2.3.2
traverse(A,B)
if A and B do not overlap **then**
  return
end if
if A and B are leaves **then**
  return intersecting leaves (DOPs)
else
  for all children A[i] and B[j] **do**
    traverse(A[i],B[j])
  end for
end if
Active Pairs method as detailed in section 2.3.1, while picking the active pairs out of the intersecting leaves of A and B.

2.3.3 Results

In order to motivate the trade-off between detection quality and speed, we first experimentally show that a detection ratio below 100% is sufficient to ensure a stable simulation. Therefore, we use the bounding volume hierarchy method (100% detection ratio) and carried out the collision response only for randomly selected collisions. Even for the very challenging simulation of cloth we experimentally found that a response (detection) ratio as low as 20% can be enough (Figure 2.10) to get good visual results. If the detection ratio is further reduced artifacts become visible.

Please note, as not all collisions are detected, robustness is a major question for stochastic collision detection schemes. It is important to ensure that the collision response does not suddenly react by inducing a high amount of potential energy which would make the system unstable. When computing the reaction to a collision we thus consider edges and triangles as inelastic rigid bodies with non-zero thickness as will be detailed in Chapter 3. The velocities are handled using the impulse-based collision response, while intersections are solved using the same method extended to positions. Due to object stiffness and damping, position and velocity corrections are then partly propagated to neighboring elements. A sound collision response is thus obtained in areas where a significant number of collisions is detected. Additionally, to handle intersections we are able to apply recent methods for untangling cloth [26].

In order to show that for a static scene the detection ratio of our presented method rapidly converges to 100% we plotted it over the number of the time step (Figure 2.11). Our new method converges much faster than the pure stochastic

Figure 2.10: Dressed woman (avatar 53000 faces, dress 3800 faces) with 100%, 20% and 10% collision response ratio.

method, because all randomly picked sample pairs are already close to local minima. To achieve the same speed of convergence with the pure stochastic method, many more random pairs are necessary.

To measure the performance of the hierarchy accelerated stochastic collision detection, we compared our method to two other approaches: to a simple stochastic approach as described in Section 2.3.1 without hierarchy acceleration and to an efficient hierarchical bounding box method based on k-DOPs as described in Section 2.3.2. As a test framework the TüTex cloth simulation system was used. For a dressed woman (Figure 2.10) and a dressed man the combined hierarchy accelerated stochastic collision detection is much faster than the pure bounding volume hierarchy or the stochastic method (Table 2.2 and 2.3). This is because it is not necessary to recompute all collisions in every time step. Instead, we obtain temporal coherence by keeping a list of collisions from step to step. Beyond that, the stochastic method is directly employed on the colliding primitives (vertices, edges, triangles) and no further computation is needed as for the pure hierarchical method. Additionally, as detailed in Section 2.3.2, the lazy hierarchy update

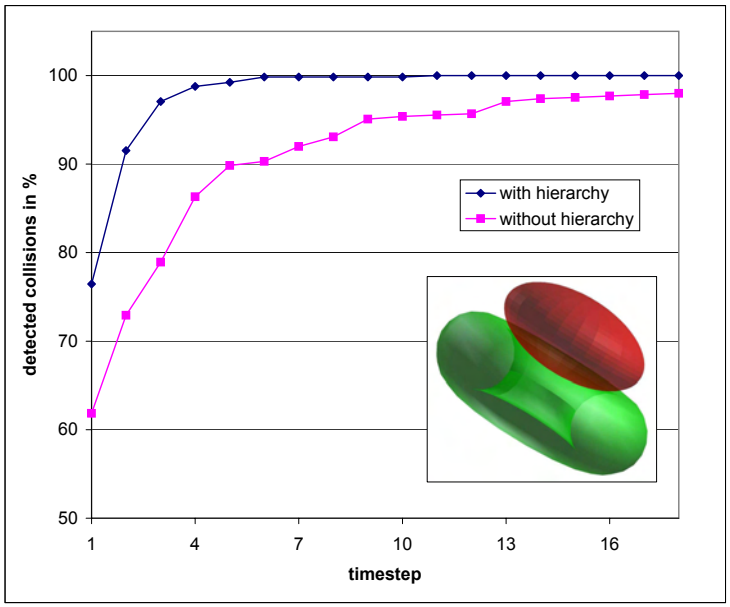
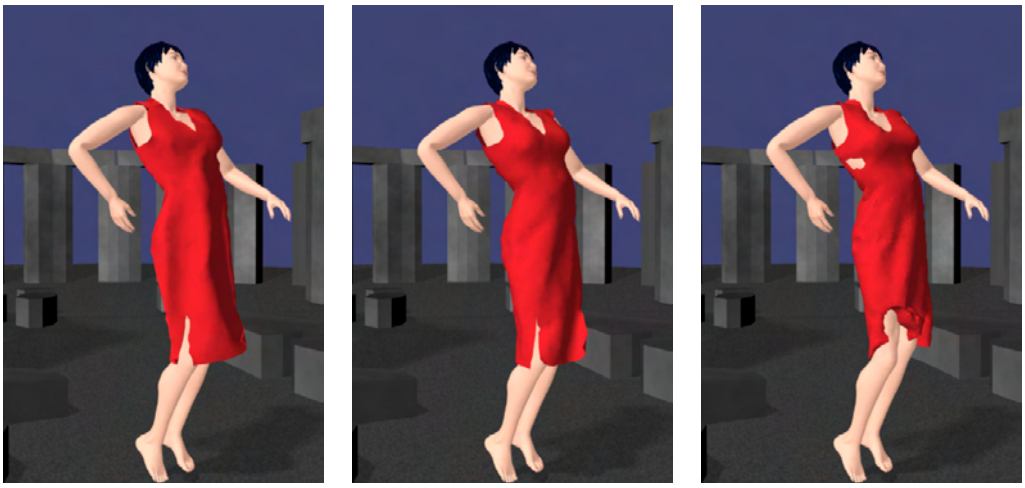


Figure 2.11: Detection ratio plotted against time step for the collisions between a static torus and ellipsoid. With the same number of randomly created candidate pairs per time step, the hierarchy accelerated stochastic method converges much faster than the pure stochastic method.

is faster and the detection of colliding DOPs can be faster as more faces are put into a DOP and therefore fewer DOPs are in the hierarchy. Compared to the pure stochastic method the combined framework achieves a much higher detection ratio with the same number of random samples, because they are only generated in "definitely close" areas. We found out that in our set-up the pure stochastic method needs about three times as much random samples to achieve the same detection ratio.

In order to test the stability of our method, we challenged it with a complex catwalk animation. We computed the dynamic scene with two different settings of our method, once with 100 new active pairs per time step and once with 50 new active pairs per time step. During the animation we achieved collision detection ratios between 40% and 80% (Figure 2.12(b)) respectively 20% and 70% (Figure 2.12(c)). For both settings the catwalk simulation remains stable during the complete 16 animation seconds (25 frames per second, time step $0.01s$). Single frames of this simulation are compared to the results of a collision detection with a pure BVH (Figure 2.12(a)) and a detection ratio of 100%. The corresponding calculation times are compared in Table 2.4 and show that without a significant

loss of quality a speed-up of a factor two is possible. With a loss in quality an even larger speed-up was achieved while the animation still remains stable. A complex collision and self-collision situation showing a piece of cloth gliding through a torus is presented in Figure 2.13. The collision detection ratio for this stable simulation lies between 30% and 70%. All measurements of computation times in this section were made on a 2.4 GHz Pentium IV with 1GB RAM.



(a) (b) (c)

Figure 2.12: Comparison of different collision detection approaches for catwalk animation (avatar 53000 faces, dress 3800 faces, 1600 time steps): calculated with a bounding volume hierarchy and 100% detection ratio (a), hierarchy accelerated stochastic collision detection with 100 new active pairs per time step resulting in detection ratios between 40% and 80% (b) and with 50 new active pairs per time step resulting in detection ratios between 20% and 70% (c).

Collision detection setup	k-dops	active pair method	combined method
Total Collision Detection	31.8	91.5	16.2
Hierarchy Update	6.8	0	0.9
Detection of colliding DOPs	11.1	0	5.3

Table 2.2: Collision detection times for the dressed woman (avatar 53000 faces, dress 3800 faces) measured in seconds for the simulation of 100 time steps.

Compared to the approach of Raghupathi et al. [109], where only pairs of edges are considered, we generalize the detection to a variety of geometric primitives: combinations of vertices, edges and triangles. The only requirement is that there exists a metric for a distance between the two primitives as e.g. edge-edge or vertex-triangle distances. In addition, voxels could also be used for col-

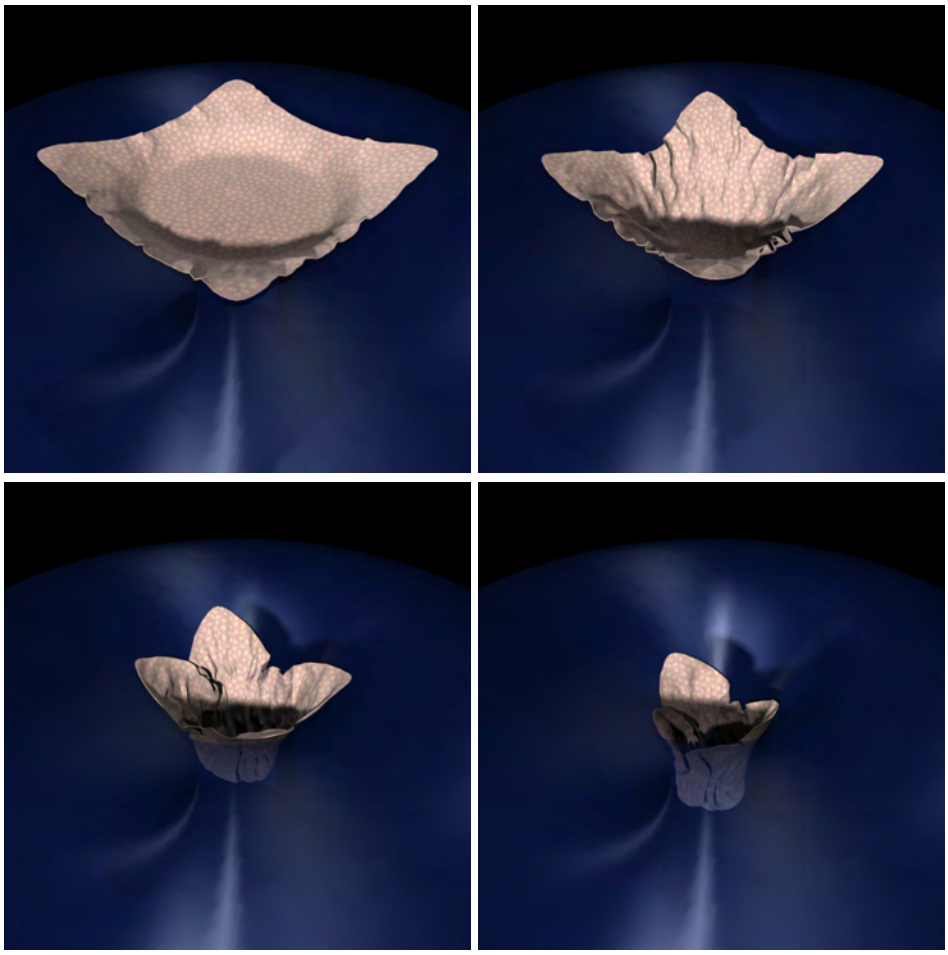


Figure 2.13: Piece of cloth gliding through a torus. The complex collisions and self-collisions were detected using the hierarchy accelerated collision detection.

lision detection between volumetric objects. Furthermore, the presented method significantly speeds up simulations and can, for example, be employed in a rapid-prototyping environment for cloth or in a real time simulation of tissue. The active pairs approach allows tuning of the computation time by limiting the number of geometric elements processed at each time step. A significant speed-up has been obtained with respect to an efficient method based on bounding volume hierarchies. Conversely, the bounding volume hierarchy approach provides us with a convenient way to initialize relevant active pairs.

Though this framework has always shown to be faster than pure hierarchical methods and yielded stable collision results for all tested examples, we have to mention that there is no guarantee that the simulation is always stable for a certain

Collision detection setup	k-dops	active pair method	combined method
Total Collision Detection	16.5	21.3	5.8
Hierarchy Update	1.9	0	1.3
Detection of colliding DOPs	4.4	0	2.3

Table 2.3: Collision detection times for the dressed man (avatar 25000 faces, shirt 4400 faces, trousers 4000 faces) measured in seconds for the simulation of 50 time steps.

Collision detection setup	k-dops	100 active pairs	50 active pairs
Total Collision Detection	395	198	165
Hierarchy Update	73	35	35
Detection of colliding DOPs	110	61	61

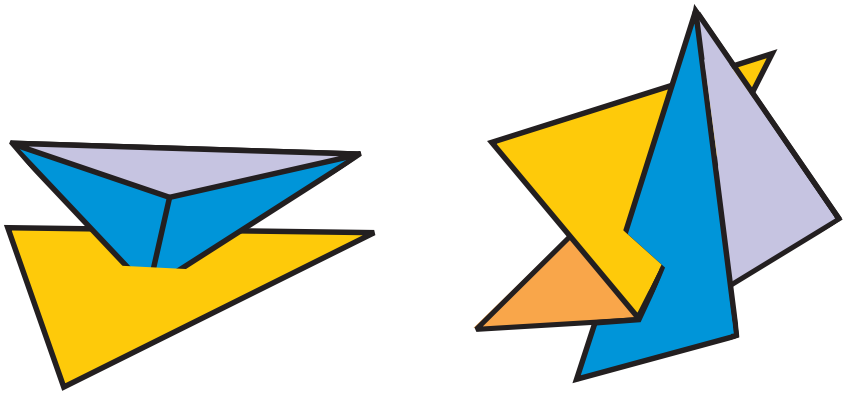
Table 2.4: Collision detection times for the catwalk animation (avatar 53000 faces, dress 3800 faces) measured in seconds for the simulation of 1600 time steps.

collision ratio, and there is no absolute guarantee to obtain a certain collision ratio for a specified number of active pairs (as with any other non-exact method). It is therefore particularly suited for interactive applications like rapid-prototyping environments.

2.4 Continuous Collision Detection

Particularly for large time steps or rapid movements it is necessary to assure a stable collision response or to maintain a collision-free state. If collisions are only detected at discrete time steps they can easily be missed and intersections may occur. The continuous collision detection not only allows to detect a collision between two consecutive steps, but also to calculate the exact time of collision. Hence, the presented approach detects collisions independently of the temporal resolution of the simulation. Furthermore, it can be employed for collision and self-collision detection of any triangular mesh. To accelerate the continuous collision detection it is combined with a bounding volume hierarchy. In a first collision detection step the bounding volume hierarchy is employed to sort out collision candidates and only returns primitive pairs that might collide between two consecutive time steps. Therefore, the bounding volumes enclose both the positions at the beginning and at the end of the time step [72, 33, 110].

In this thesis we mainly employ continuous collision detection to calculate contact points and avoid intersections caused by subdivision and wrinkles simula-



(a) (b)

Figure 2.14: The two possible collision types between triangles: vertex-triangle collision (a) and edge-edge collision (b).

tions (Chapter 4 and 5). This section partially follows our work in [6, 70].

2.4.1 Configurations of Colliding Triangles

As triangle meshes are the preferred representation of cloth simulations, we confine ourself to the continuous collision detection between triangle-triangle pairs. The candidate pairs are returned by a bounding volume hierarchy that is used as a first step of the collision detection process. Let t_i be the initial time, where two corresponding triangle meshes are collision-free. During the time interval $[t_i, t_i + \Delta t]$, representing one time step, the vertices of these two meshes move and collisions between the triangles of the meshes may occur. We suppose that for each vertex the initial position and velocity at time t_i is given. The new positions of the vertices at time $t_i + \Delta t$ can easily be calculated by assuming uniform motion, i.e. all vertices move with a constant velocity [97, 106].

In general, two triangles can collide in two different ways. Either a triangle vertex of the first triangle collides with the other triangle (Figure 2.14 (a)) or two edges of the different triangles intersect (Figure 2.14 (b)). To detect all possible collisions between two triangles it is therefore necessary to check six vertex-triangle combinations und nine edge-edge combinations for intersection. Each of the two triangles has to be tested against all vertices of the other triangles and all edges of one triangle have to be checked against all edges of the other triangle.

2.4.2 Time of Collision

To detect collisions between a vertex and a triangle or between two edges in motion, it is necessary to find the time within the interval $[t_i, t_i + \Delta t]$ at which all four involved vertices are coplanar. For a collision between the two primitives, this is a necessary condition. We focus on the sufficient conditions in Section 2.4.3 and 2.4.4.

For the necessary condition of coplanarity, let $\vec{v}_1, \vec{v}_2, \vec{v}_3, \vec{v}_4$ be the constant velocities during the interval $[t_i, t_i + \triangle t]$ of the four moving points P_1, P_2, P_3 and P_4. Their respective position vectors are given by:

$$\begin{aligned}
\vec{x}_1(t) &= \vec{x}_1(t_i) + (t - t_i)\vec{v}_1 \\
\vec{x}_2(t) &= \vec{x}_2(t_i) + (t - t_i)\vec{v}_2 \\
\vec{x}_3(t) &= \vec{x}_3(t_i) + (t - t_i)\vec{v}_3 \\
\vec{x}_4(t) &= \vec{x}_4(t_i) + (t - t_i)\vec{v}_4
\end{aligned}$$

The difference vectors $\overrightarrow{x_1x_2}(t), \overrightarrow{x_1x_3}(t)$ and $\overrightarrow{x_1x_4}(t)$ build a parallelepiped volume. At the time $t \in [t_i, t_i + \Delta t]$ for which this volume is zero, all four points are coplanar. This results in finding the roots of the following cubic equation:

$$(\overrightarrow{x_1x_2}(t) \times \overrightarrow{x_1x_3}(t)) \cdot \overrightarrow{x_1x_4}(t) = 0 \tag{2.1}$$

Equation (2.1) gives a maximum of three real roots. Any root outside $[t_i, t_i + \Delta t]$ is discarded, the remaining roots are checked if they fulfill the sufficient condition to result in a collision. If collisions at several times t are found, only the first one is handled, the others are discarded.

2.4.3 Vertex-Triangle Collision

For a collision between a vertex and a triangle (Figure 2.15 (a)) the sufficient condition is that the vertex lies within the triangle. Hence, first it is checked if the vertex P_4 is closer than a tolerance value ε to the plane containing the triangle $P_1P_2P_3$ with the normal $\vec{n}$: $|\overrightarrow{x_1x_2}(t) \cdot \vec{n}| < \varepsilon$. Then the vertex P_4 is projected onto that plane and the barycentric coordinates w_1, w_2, w_3 relative to the triangle are calculated using Equation (2.2). If all barycentric coordinates lie within the

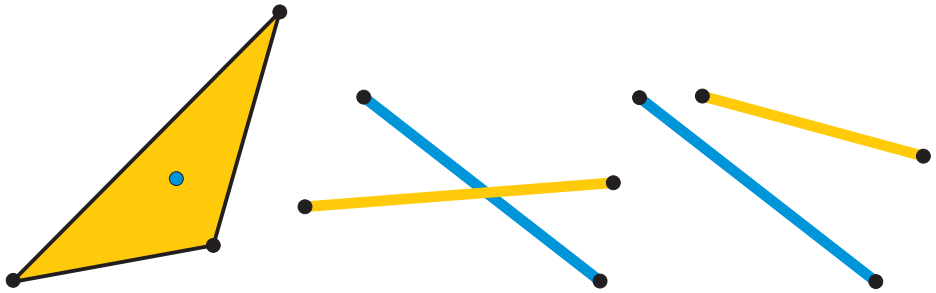


Figure 2.15: Different collision configurations: vertex-triangle collisions (a), warped edges with closest point within both edges (b) or at the endpoint of an edge (c).

interval $[0, 1]$ and if $w_1 + w_2 + w_3 = 1$, then a collision occurs.

$$\begin{pmatrix} \overrightarrow{x_1x_3}(t) \cdot \overrightarrow{x_1x_3}(t) & \overrightarrow{x_1x_3}(t) \cdot \overrightarrow{x_2x_3}(t) \\ \overrightarrow{x_1x_3}(t) \cdot \overrightarrow{x_2x_3}(t) & \overrightarrow{x_2x_3}(t) \cdot \overrightarrow{x_2x_3}(t) \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} \overrightarrow{x_1x_3}(t) \cdot \overrightarrow{x_4x_3}(t) \\ \overrightarrow{x_2x_3}(t) \cdot \overrightarrow{x_4x_3}(t) \end{pmatrix} \tag{2.2}$$

The tolerance value ε can be used to account for rounding errors or to take the cloth thickness into account. Bridson et al. [33] therefore assume a collision if all barycentric coordinates lie within the interval $[-\varepsilon, 1 + \varepsilon]$. They choose the constant ε as the cloth thickness divided by the average edge length.

2.4.4 Edge-Edge-Collision

To detect colliding edge-edge pairs that fulfill Equation (2.1) at a specific time t the two points, one on each of the two edges $\overline{P_1P_2}$ and $\overline{P_3P_4}$ that are closest are found and their euclidian distance is checked. Is this distance smaller than a specified tolerance value δ the two edges collide at time t.

The search for the two closest points starts by checking for the degenerate case of parallel edges, i.e. the cross product $|\overrightarrow{x_1x_2}(t) \times \overrightarrow{x_3x_4}(t)|$ is smaller than a round-off tolerance. If not, the points lying on the skewed infinite lines that are closest to each other are found by solving Equation (2.3).

$$\begin{pmatrix} \overrightarrow{x_1x_2}(t) \cdot \overrightarrow{x_1x_2}(t) & -\overrightarrow{x_1x_2}(t) \cdot \overrightarrow{x_3x_4}(t) \\ -\overrightarrow{x_1x_2}(t) \cdot \overrightarrow{x_3x_4}(t) & \overrightarrow{x_3x_4}(t) \cdot \overrightarrow{x_3x_4}(t) \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \overrightarrow{x_1x_2}(t) \cdot \overrightarrow{x_1x_3}(t) \\ -\overrightarrow{x_3x_4}(t) \cdot \overrightarrow{x_1x_3}(t) \end{pmatrix} \tag{2.3}$$

If the resulting points with the position vectors $x_1(t) + a\overrightarrow{x_1x_2}(t)$ and $x_3(t) + b\overrightarrow{x_3x_4}(t)$ are on the finite edges, the points with the smallest distance are found and the process is finished (Figure 2.15 (b)). If these two points are not on the finite edges, they are moved to the closest endpoints of the edges. The point that moved the most during this process is one of the two points with the smallest distance. The second point is found by projecting the first onto the second infinite line and if this point is not lying on the finite edge moving it to the closest endpoint (Figure 2.15 (c)).

2.5 Summary

As collision detection is a substantial part in the simulation of virtual cloth, in this chapter we presented the state of the art of collision detection in this context. We particularly exposed the advantages and disadvantages with respect to topology changes, balance between speed and quality, self-collision detection and animated avatars. In this regard, we are the first who gave the developers of cloth simulation systems a decision matrix in order to ease the process of choosing the best method for a specific application. For the use in our cloth simulation system TüTex we showed that bounding volume hierarchies are the best choice, which, however, do not allow to balance speed and quality of the collision detection process.

To overcome this drawback, we presented a new collision detection method, the combination between bounding volume hierarchies and a stochastic collision detection approach. As with this approach the sample pairs for the stochastic collision detection are only generated in areas that were found close by the hierarchical collision detection, the hierarchy accelerated stochastic collision detection shows a much better performance than both other approaches on their own. To motivate the use of stochastical collision detection approaches, we have shown that in order to obtain stable cloth simulations no detection ratio of 100% is necessary. To accelerate the collision detection we exploit temporal coherence between time steps and reduce the update costs of the bounding volume hierarchy by a lazy hierarchy update. We see the strength of these methods particularly in rapid-prototyping or real time applications.

To detect collisions that occur between the single time steps, we presented an algorithm for continuous collision detection. Particularly for the post-processing methods of cloth meshes detailed in Chapter 4 and 5 continuous collision detection is used in order to find penetrations caused by subdivision or geometric wrinkle generation.

CHAPTER 3

Collision Response

3.1 Introduction

As soon as a simulated piece of cloth is close enough to another object or to parts of itself, an interaction takes place that has to be treated in order to maintain a realistic simulation. This response to detected collisions or proximities is used to avoid interpenetrations of the objects and to model kinetic and static friction.

Not only for collision detection there is a great difference between algorithms for rigid and deformable objects. Also for collision response the special properties of highly deformable objects like cloth have to be taken into account. The main difference to collision response for rigid objects is the number of collisions that occur at a time. While rigid objects usually collide only at a few points (apart from resting contacts), deformable materials might be in contact in huge areas leading to a large number of collisions. The best example for this scenario is a garment worn by a person, which is very close to the body at almost any region. The second reason complicating collision response for cloth is the extremely small thickness. Even marginal interpenetrations can therefore cause visually disturbing artifacts and lead to difficulties in resolving the collisions as the order and orientation of the objects is violated. As deformable surfaces often are represented by polygonal

meshes, and collisions are detected between the mesh primitives, another problem is the distribution of the collision response to the individual vertices of the mesh.

For the collision response step in the simulation pipeline, several methods were proposed in literature. They can be classified as physical and geometrical approaches. Collision response schemes that model the repulsive force field between the objects are called physical methods. Approaches which constrain the degrees of freedom of the body motion, or which directly modify positions, velocities or even accelerations are referred to as geometric response schemes. In the following, we first discuss these two classes of collision response approaches employed in cloth simulation. Then, we detail the requirements for the collision response schemes utilized in TüTex (Section 3.2). We explain the realization of a constraint based collision response in our simulator. As this response scheme alone is not capable of resolving collisions for dynamic rigid objects and has difficulties with the resolution of self-collisions we introduce a scheme that changes the position and velocity of the animated vertices in order to perform a stable collision response. The problem of distributing the collision response for polygonal meshes on the individual vertices while respecting physical conservation laws is also examined in this section. Our results using these collision response schemes are presented and discussed in Section 3.3.

3.1.1 Physical Collision Response

The idea behind physical collision response is to model the repulsive potential force field F between solid objects of distance r with $F \sim r^{-12}$, which avoids intersection in reality. As forces can directly be integrated into the mechanical model and the arising differential equations can be solved accordingly, physical response schemes are the more formal approach to collision handling. Though force field based collision response schemes have the advantage of being very close to the underlying physical laws, they suffer from a couple of major disadvantages. The most important challenge when employing force fields is to model the contact between the colliding objects, i.e. to assure a minimal collision distance. Modeling the very short-ranging force fields directly is infeasible using the large time steps of $10^{-3}s$ to $10^{-1}s$ usually used in numerical solvers for cloth simulations. For this case, the very strong and rapidly declining potential cannot model the collision interaction as it may completely be passed between two time steps. Hence, it is necessary to model the force field with longer ranging potential fields. This, however, can cause unrealistic simulation results, as the assured minimal distance between colliding objects is too large. Additionally, force fields always induce stiffness into the differential equations reducing the performance

and stability of the numerical integrators [25, 50]. Despite these disadvantages force fields are used for collision response situations where small time steps are employed or where the minimal distance between the colliding objects is not that important like for self-collisions in dynamic scenes. To avoid an unnatural sliding of objects over each other, forces can easily be used to model Coulombian friction.

Moore and Wilhelms [97] proposed the use of repulsive springs for separating simulated rigid objects when they get too close. Terzopoulos and Fleischer [119] keep a continuous formulation by using a deflective force field around deformable collision objects. For cloth simulations Lafleur et al. [85] proposed a force field around each colliding triangle in which the strength of the force field depends on the velocity, the normals and the distance of the colliding objects. Baraff et al. [25] use strong spring forces to avoid self-collisions by pushing cloth objects apart while damping the movement to avoid bouncing. A hybrid approach combining a controlled force field with a geometric collision response method is proposed by Bridson et al. [33]. They limit the strength of the collision forces to avoid inducing stiffness into the differential equation. In order to resolve complicated self-collision situations, Baraff et al. [26] use forces between the involved particles. Areas of the cloth that have intersected other areas and are located on the wrong side are separated using attractive forces, while close particles that are still on the correct side repel each other.

3.1.2 Geometric Collision Response

Geometric collision response tries to simulate the effect of the underlying physical principles without explicitly using repulsive forces. The proposed algorithms directly change the geometrical state, i.e. position, velocity or acceleration, of the simulated cloth or constrain the movement of the object into certain directions. The main advantage of this approach is the separation of collision response and numerical integration. Hence, the performance of the differential equation solvers is not reduced due to induced stiffness. Volino and Magnenat-Thalmann [133] point out that this separation is also the biggest drawback, because nothing ensures the compatibility between the change of the geometric state and the correct result of a repulsive force field. Also the conservation of energy and momentum should be respected when manually manipulating the system state. Furthermore, the geometric response to several interacting collisions has to be compatible, i.e. it has to be assured that the response to one collision does not cause new collisions.

Eberhardt et al. [48] proposed a method to alter positions and velocities based on a ray intersection test. If the ray from the initial position to the new position after the time step intersects an obstacle, the surface normal at the intersection

point is calculated. Using this surface normal they move the cloth particle to a point laying a minimum distance above the obstacle. Additionally, the velocity is altered, preventing the particle from moving any further in direction of the obstacle. Using this velocity modification the friction and elasticity of the contact are also modeled. Provot [106] used a similar approach of modifying the velocities of the simulated particles to model friction and repulsion forces.

An early realization of constraints in the simulation of deformable objects was presented by Platt and Barr [104]. They use "reaction constraints" for allowing animators to control the motion and to avoid interpenetration between the simulated objects. In the work of Baraff et al. [25, 71] small position changes are fed back to the differential equations while enforcing constraints in the direction of the position change by using a modified conjugate gradient method. They state that this approach avoids visible artifacts due to position alterations, e.g. a jumpy behavior, by using the implicit Euler method as a filter. Volino and Magnenat-Thalmann [131, 132, 133] propose an algorithm to alter positions, velocities and accelerations depending on the distance of the colliding objects. Additionally, strategies are given how the orientation of the objects relative to each other can be maintained without using a continuous collision detection scheme. Bridson et al. [33] employ a combined approach using repulsive forces as well as velocity changes due to added impulses. Objects closer to each other than a certain threshold are separated using the force field while objects intersecting each other during a time step are handled using the impulses. Using this approach they claim to significantly reduce the total number of collisions that need to be treated.

In the context of the geometrical methods, strategies to respond to interacting collisions have been proposed. Provot [106] came up with the idea of clustering self-collisions together into "zones of impact" in order to treat them as a whole. Provot motivates this idea with the observation that these multiple collisions can hardly move relative to each other. Bridson et al. [33] build on this idea and include it into an iterative solving strategy. They have shown that a simultaneous correction scheme that combines all calculated collision responses and adds them to the particles using weighting factors can give convincing results. Volino and Magnenat-Thalmann [131] propose to use a linear equation system in order to treat all collisions globally. In order to avoid discontinuities of the collision response due to the discretization of the objects, Heidelberger et al. [67] compute the direction of the response forces based not only based on a pair of primitives but on a larger set of surface features.

3.2 Collision Response Schemes for TüTex

After giving this general overview of collision response schemes, we now discuss the methods employed within TüTex. To yield a sophisticated response for our system, these methods need to meet the following requirements:

- Provide a stable collision response for both dynamic and static scenes, e.g. standing or walking avatars, even for large time steps.
- Resolve self-collisions and collisions with environment objects.
- Respond to collisions between different layers of cloth.
- Distribute the collision response to the mesh vertices.

For performance reasons large time steps are employed in TüTex. Therefore geometric collision response schemes are preferred. Physical schemes would require long ranging force fields leading to visually disturbing gaps between cloth and body.

In the remainder of this chapter we first discuss how the effects of the proposed geometric collision response between cloth meshes can be distributed to the individual vertices. Then, in Section 3.2.2 a constraint-based collision response will be presented. This scheme is a modification of the method proposed by Baraff and Witkin [25]. As this response method shows problems in combination with dynamic rigid objects, we present a collision response approach which modifies the system velocities by exerting impulses on the cloth in Section 3.2.3. Using this method both collisions and self-collisions can be treated correctly, which yields a stable response for static and dynamic scenes.

3.2.1 Geometric Response for Triangular Meshes

Geometric collision response schemes for triangular meshes often calculate the response, i.e. alteration of position or velocity, for two points of the corresponding cloth with a minimum distance to each other. For colliding edges and triangles, however, the closest distance is not necessarily at the vertices of the primitives. Because the collision response, e.g. an impulse, cannot be applied directly to internal points, it has to be distributed to the corners of triangles or the endpoints of edges.

In addition, to calculate the collision response it is often necessary to compute the velocity of a internal point of a triangle or an edge. To accomplish this, we use

linear interpolation between the corner points of edges or triangles. The velocity $\vec{v}$ of the inner point P_4 of a triangle $P_1P_2P_3$ is calculated by using its barycentric coordinates w_1, w_2 and w_3 as $\vec{v} = w_1\vec{v}_1 + w_2\vec{v}_2 + w_3\vec{v}_3$. Here, we denote as $\vec{v}_1$, $\vec{v}_2$ and $\vec{v}_3$ the velocities of the three corner points. For a point laying on a fraction a of an edge $\overline{P_1P_2}$ the interpolated velocity is $\vec{v} = (1-a)\vec{v}_1 + a\vec{v}_2$.

The collision response applied to the colliding objects has to fulfill basic mechanical laws. The most important ones are Newton's axioms and the consequential conservation of impulses. Additionally, for the collision response, the continuity across borders of the mesh elements is necessary. Based on these principles, we can deduce the velocity changes for the corner points of triangles or the endpoints of edges. First we show how the velocities of a vertex-triangle collision between point P_4 and triangle $P_1P_2P_3$ need to be corrected. To counteract the collision, we want to apply an impulse I in direction $\vec{n}$ (usually the normal direction of the triangle) to the two cloth points of a vertex-triangle collision (i.e. $I\vec{n}$ to the vertex, $-I\vec{n}$ to the point within the triangle). For the collision response, we want the relative velocity $\vec{v}_{relN}$ in normal direction to vanish:

$$\vec{v}_{relN} = 0 = \vec{v}_4 - \omega_1\vec{v}_1 - \omega_2\vec{v}_2 - \omega_3\vec{v}_3. \tag{3.1}$$

As we want to assure continuity when a collision moves from one triangle to a neighboring one, we distribute the impulses by weighting them with the barycentric coordinates. The resulting velocity changes are then given by:

$$\Delta\vec{v}_1 = -\frac{\omega_1}{m_1}I^*\vec{n}; \quad \Delta\vec{v}_2 = -\frac{\omega_2}{m_2}I^*\vec{n}; \quad \Delta\vec{v}_3 = -\frac{\omega_3}{m_3}I^*\vec{n}; \quad \Delta\vec{v}_4 = \frac{1}{m_4}I^*\vec{n} \tag{3.2}$$

where m_i is the mass of the corresponding vertex.

By inserting Equation (3.2) in (3.1) we get the impulse I^* that needs to be applied:

$$I^*\vec{n} = \frac{\vec{v}_{relN}}{\frac{1}{m_4} + \frac{\omega_1^2}{m_1} + \frac{\omega_2^2}{m_2} + \frac{\omega_3^2}{m_3}}.$$

For the edge-edge collision, where a point with relative position a along the edge $\overline{P_1P_2}$ interacts with a point with relative position b along the edge $\overline{P_3P_4}$, the velocity corrections are given by:

$$\Delta\vec{v}_1 = \frac{a}{m_1}I^*\vec{n}; \quad \Delta\vec{v}_2 = \frac{1-a}{m_2}I^*\vec{n}; \quad \Delta\vec{v}_3 = -\frac{b}{m_3}I^*\vec{n}; \quad \Delta\vec{v}_4 = -\frac{1-b}{m_4}I^*\vec{n} \tag{3.3}$$

with the impulse

$$I^*\vec{n} = \frac{\vec{v}_{relN}}{\frac{a^2}{m_1} + \frac{(1-a)^2}{m_2} + \frac{b^2}{m_3} + \frac{(1-b)^2}{m_4}}$$

If the mesh points should be set apart, we alter the vertex positions by distributing the displacements according to the proposed method for velocity changes.

3.2.2 Constraint-based Collision Response

Constraint-based collision response schemes restrict the possible movement of cloth vertices to directions orthogonal to a constraint vector. We realize the velocity constraints by projecting the state vector of the velocities into the allowed subspace after each matrix operation [50]. This projection is done by using the matrix $P := Id - \sum c_i c_i^T$, where Id is the identity matrix and c is the state vector of the normalized constrained direction. The constrained directions are derived from the collision situations between vertices and triangles, or edges and other edges, respectively. For vertex-triangle collisions, the vertices are constrained in direction of the triangle normal, for edge-edge collision, the vertices are constrained in the direction of the shortest distance.

While the constraints prevent a further interpenetration between the colliding objects, it is also necessary to reset the vertices to a target position in order to resolve the interpenetration completely. This target position is chosen to assure a minimum distance between all colliding objects. If a vertex of a cloth mesh gets closer than this minimum distance to the collision object, the vertex velocity is set to move it away from the collision object until this distance is assured. As soon as the vertex departs from the collision object or when it is dragged away from it by internal or external forces, the constraint must be released. To avoid bouncing effects due to repeated enforcing and releasing of constraints, it is our experience that it is better to gradually move the vertex towards the minimum distance instead of doing this in one simulation step.

Using constraints for collision response is successful as long as only collisions between the cloth and a static rigid object are considered. When the rigid object moves or when more self-collisions need to be treated correctly, the limitations of this method become apparent. The main disadvantage is the necessity to explicitly release the constraints after the collision has been resolved and the objects move away from each other, which proves to be infeasible in connection with animated rigid objects or complicated self-collision situations and results in sticking between different layers of cloth. To overcome these limitations we propose a collision response using a direct alteration of velocities and positions based on the

laws of impulse conservation.

3.2.3 Impulse-based Collision Response

The idea behind impulse-based collision response methods is to regard the collision between colliding objects as impact, where impulses in opposite directions are applied on the involved mesh points in order to prevent interpenetration. An approximation of the complex behavior during the collision of objects with deformation and restitution is given by Poisson's hypothesis [96]. Using Poisson's hypothesis for collisions of cloth, the material parameter for the restitution usually is assumed to be zero [33], stating that no energy is stored in the deformation during the collision process. So, interactions of cloth objects are usually modeled assuming completely inelastic collisions [106, 33, 43].

In our cloth response model we also model completely inelastic collisions. Therefore, if two points inside the colliding mesh primitives are approaching each other with the relative velocity in normal direction $\vec{v}_{relN}$ and are closer than the collision distance, an impulse is applied to avoid the interpenetration. The required impulse as well as the velocity of points inside primitives is calculated using the relations given in Section 3.2.1. Also, the respective relations are used to distribute the impulse to the vertices of the mesh primitives. If the two colliding points are already closer than a use- defined threshold, e.g. the cloth thickness, the two primitives are additionally set to this minimum distance to resolve the collision.

To avoid the visually disturbing sliding of cloth on static objects we model friction forces using Coulombs law. The relative velocity of the two objects in contact as $\vec{v}_{rel}$. Its component in tangential direction is denoted with $\vec{v}_{relT}$. Let $\vec{F}_N$ and $\vec{F}_T$ be the normal and tangential frictional force. Then the laws of friction are given by:

$$|\vec{v}_{relT}| \neq 0 \Rightarrow \vec{F}_T = -\mu|\vec{F}_N|\frac{\vec{v}_{relT}}{|\vec{v}_{relT}|} \tag{3.4}$$

$$|\vec{v}_{relT}| = 0 \Rightarrow |\vec{F}_T| < \mu|\vec{F}_N| \tag{3.5}$$

Here, we denote the friction coefficient as μ. While Equation (3.4) models a sliding contact with friction, Equation (3.5) models a sticking contact without relative motion. Hence, for $\mu = 0$ there is sliding without friction, while for $\mu = \infty$ there is no sliding at all. To adapt these macroscopic laws to the situation of colliding primitives, simplifications concerning the normal and tangential forces

have to be made. For constant time steps, we state that the forces $\vec{F}_N$ and $\vec{F}_T$ are proportional to the respective relative velocities. In this way friction is modeled using micro-collisions as discussed by Bridson et al. [33] and Provot [106].

3.3 Results

We first implemented the constraint-based collision response into TüTex and tested it with various collision situations. In Figure 3.1, a piece of cloth (4000 triangles) is falling on a plane, which leads to collisions between the textile and the static rigid object as well as to self-collisions. In this scene the constraint-based collision response resolves both kinds of occurring collisions.

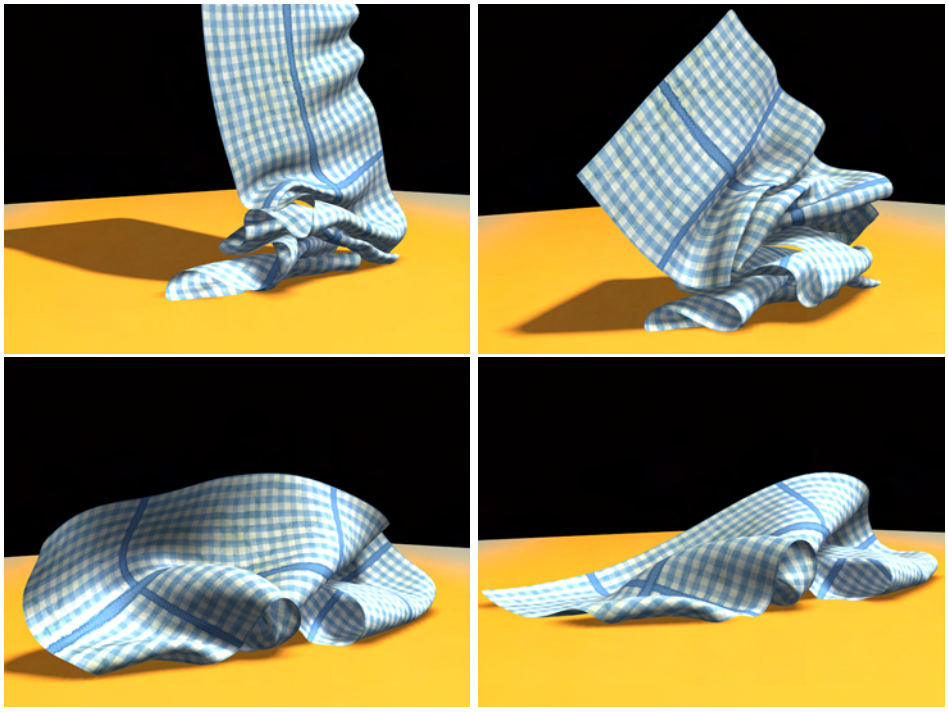


Figure 3.1: Different frames of an animation of a ribbon (4000 triangles) falling on a table resulting in complex collisions and self-collisions. The collisions and self-collisions in this scene were resolved using the constraint-based response.

For dynamic rigid objects, however, where the piece of cloth is entrained, the constraints cannot be released correctly and the cloth gets entangled. Therefore, for these scenes we employ the impulse-based collision response. Using this response scheme, we are able to resolve collisions and self-collisions. Additionally,

static and kinetic friction are modeled, as demonstrated in Figure 3.2. In this animated scene, a piece of cloth is draped over a rotating sphere, which drags the cloth causing complex folds and numerous collisions and self-collisions.

Figure 3.2: Simulation of a piece of cloth draped over a rotating sphere. The collisions and self-collisions in this animation were resolved using the impulse-based response.

3.4 Summary

To resolve collisions in cloth simulations a collision response scheme needs to be applied. This collision response scheme has to provide a stable response for dynamic and static scenes, has to resolve both collisions and self-collisions, and should be able to distribute the collision response to the mesh vertices belonging to the colliding primitives. After giving an overview of physical and geometric collision response, we have presented two sophisticated methods to resolve detected collisions in TüTex in this chapter. The constraint-based method restricts the possible movement of cloth vertices to directions orthogonal to a constraint vector. Using this method, we obtained convincing results and are even able to treat complex self-collisions correctly. Constrained vertices must however

be released manually when they depart from the collision object. This manual release becomes infeasible for animated rigid objects. To overcome this limitation, for TüTex we have implemented an impulse-based response scheme that is able to smoothly resolve collisions and self-collisions. To conserve the impulses and assure continuity across the borders of mesh elements, we have presented a physically-based approach to distribute the impulses to the corners of triangles and the end points of edges. Using the impulse-based method we are able to get stable results of dynamic scenes, where collisions and self collisions as well as friction are simulated correctly.

CHAPTER 4

Collision-free Subdivision

4.1 Introduction and Related Work

In the physically-based simulation of cloth, the objects, i.e. avatars but also the garment, are usually represented as polygonal meshes. Besides many advantages, like the simple modeling of different cloth patterns this also creates a difficult dilemma. In order to obtain visually pleasing animations, high resolution meshes are needed. If coarse meshes were used, the disturbing polygonal structure of the mesh would be visible. However, when these high resolution cloth meshes are simulated, the performance of numerical solvers as well as of collision detection methods decreases significantly.

This dilemma can be addressed by directly simulating smooth surfaces instead of polygonal meshes. Thingvold et al. [121] used dynamic B-splines to refine the used meshes in regions of interest. They associated the control points of the splines with the nodes of the simulation. In the paper about "Geri's Game" from Pixar, DeRose et al. [42] utilized subdivision surfaces for a physical cloth simulation including the necessary collision detection. The drawback of directly simulating smooth surfaces is, however, that simulation and representation are no longer independent causing collisions depending on the level of detail of the

underlying object. Therefore for each subdivision step a collision detection is necessary. Additionally, many simulation techniques like finite element methods need special adaptations when used with these surface representations. Hence, as the direct simulation on smooth surfaces is not applicable, we expand an idea from Bridson et al. [33] who propose the simulation of coarse meshes, which are refined in a post-processing step. While this chapter focusses on the smoothing of the triangular meshes using different subdivision schemes, the following chapter introduces a method that allows the enhancement of the meshes with folds and wrinkles.

We start by giving a introducing to interpolating and approximating subdivision schemes (Section 4.2). As the subdivision process not only refines the mesh but also alters the position of mesh vertices, collisions of the cloth mesh with other objects or with itself may occur. This is particularly true as cloth is a very thin object and mostly lies very close to the body. How these collisions can be detected and handled based on the continuous collision method will be detailed in Section 4.3. In Section 4.4 we evaluate which of the presented subdivision methods fits best to the needs of a cloth simulation. As a test scenario, we compare simulations of high resolution cloth meshes without subdivision to simulations of low resolution meshes with post-processing by subdivision. Parts of this chapter follow our work in [6, 70].

4.2 Subdivision Methods

Subdivision has become a major field in computer graphics. The idea of subdivision is to define a smooth surface as the limit of a sequence of successive refinements [112]. Practically, new vertices are added to an initial mesh and defined as affine combination of neighboring vertices. Subdivision goes back to the late 1940s when G. de Rahm used “corner cutting” to describe smooth curves. In the area of surface modeling the publications of Catmull and Clark [35] and Doo and Sabin [45] in 1978 marked the beginning of subdivision. They proposed the generalization of spline-patch methods to meshes of arbitrary topology. The importance of subdivision in computer graphics is caused by the need for multiresolution techniques. This possibility to implement level-of-detail into meshes is the major reason why subdivision is used in our work to accelerate cloth simulation.

To classify the vast variety of subdivision methods that have been proposed in the last three decades there can be given three main criteria [112]:

- Mesh topology: There exist subdivision schemes for all kinds of mesh

topologies, but only for regular quadrilateral, triangular and hexagonal meshes there are regular subdivision schemes. As all the cloth simulation algorithms in our simulator, both particle or finite element methods, work on triangular meshes, we will confine ourself to the corresponding subdivision methods.

- Refinement rule: Different refinement rules have been proposed: vertex insertion and corner cutting. Vertex insertion methods split each edge into two new ones, old vertices are kept, and new vertices inserted on edges are connected among each other. Corner cutting methods create for each old face a new face inside of it. This results in two new vertices for each old edge, and a new face for each old edge and vertex. All old vertices are discarded. For triangular meshes only regular vertex insertion methods have been described.

- Interpolating or approximating (Figure 4.1): For interpolating subdivision schemes all points of the control mesh are also points of the subdivision surface. Hence, at each recursive step, the position of the existing points is kept, a smoothing of the surface is only achieved by the insertion of new points. Using this class of subdivision schemes, it is more obvious to estimate how the limit surface will look like. Therefore, these methods are often used for modeling with subdivision surfaces. However, due to the static control points, interpolating subdivision meshes may show bulges and bumpy behavior for unsteady initial meshes. For approximating subdivision schemes the points of the controlling mesh are not part of the subdivision surface. In each subdivision step the existing vertices are moved in direction to the limit surface. The major advantage of approximating schemes is that the resulting subdivision surface tends to be very unruffled, having few undulations and ripples. Even for unsteady initial meshes these schemes yield smooth results as the sharpest points move the furthest in direction to the limit surface. If it is the intention to maintain these surface features this behavior can also be disadvantageous.

As the characteristics of interpolating and approximating subdivision schemes are very different we compare their applicability for the refinement of cloth meshes. To choose suitable candidates for the post-processing of cloth simulations, only methods at least producing C^1-continuous surfaces around regular and irregular vertices, i.e. vertices in a triangular mesh with a valence unequal six, are considered. This is important as it is not always possible to tessellate the cloth meshes in the TüTex simulator regularly, particularly under the constraint of special seam points where different cloth patterns are sewed together. Therefore, it is also important that the subdivision scheme works for any triangulations, including the

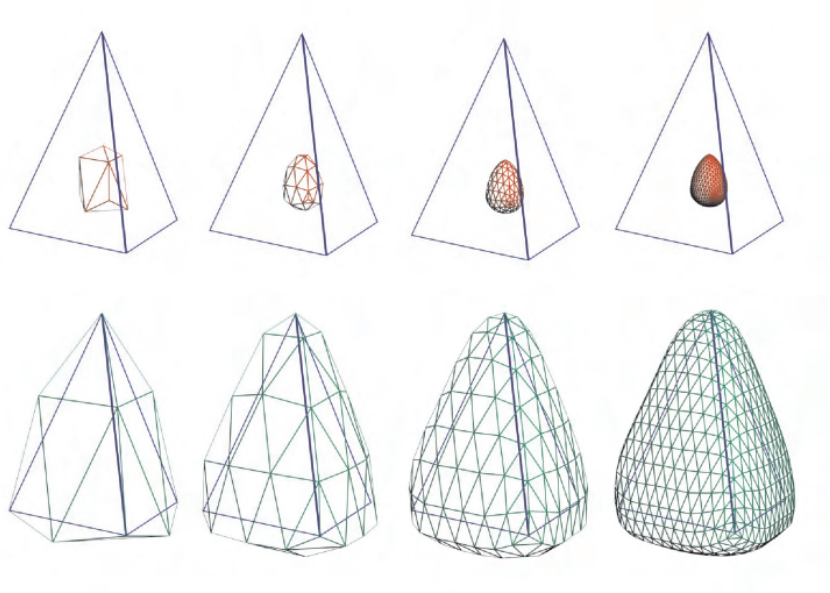


Figure 4.1: Subdivided tetrahedron: With the approximating Loop method (top row) and with the interpolating Butterfly method (bottom row).

proper handling of boundaries. Additionally, in order to obtain fast subdivision algorithms only methods that need a small neighborhood to define the new vertex positions are considered. Two of the most prominent and fast representatives of interpolating and approximating schemes producing C^1-continuous surfaces around irregular vertices are the modified Butterfly and the modified Loop method, respectively. Based on the unmodified Butterfly and Loop scheme, in the following we detail the modified methods and compare them with respect to the post-processing of cloth meshes.

4.2.1 Butterfly Subdivision

The interpolating Butterfly scheme was originally proposed by Dyn et al. [46]. It got its name by the shape of its edge mask (Figure 4.3 (a)), the rule to determine new edge points. Though this scheme is defined on arbitrary triangular meshes, for extraordinary points of valence $k = 3$ and $k > 7$ the limit surface is not C^1-continuous and therefore it can show undesirable bulges and rills for irregular meshes. By adding special subdivision rules for these extraordinary vertices, Zorin et al. [140] enhanced the Butterfly scheme to produce C^1-continuous surfaces for arbitrary meshes. As cloth meshes often contain such vertices, the modified Butterfly scheme is better suited for the application in cloth simulation

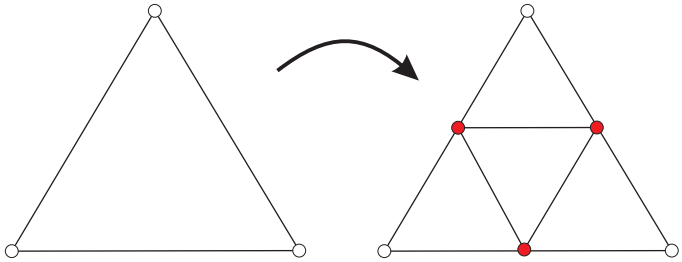


Figure 4.2: Subdivision of triangle by 1-to-4 split.

and will be detailed in the following.

Subdivision Rules: The scheme divides each face of the triangular mesh by carrying out a 1-to-4 split of the triangles (Figure 4.2). All newly created vertices are called *odd vertices*, the existing ones *even vertices*. As the Butterfly scheme is an interpolating subdivision method all even vertices stay at their positions while the odd vertices (points on edges in the original mesh) are moved corresponding to subdivision masks. Figure 4.3 shows the modified Butterfly masks for regular and irregular internal vertices with the weights given in equations (4.1) to (4.3), where k is the valence of the internal vertex.

For $k = 3$ the weights are given by:

$$\alpha = \frac{3}{4}; \quad \beta_0 = \frac{5}{12}; \quad \beta_1 = -\frac{1}{12}; \quad \beta_2 = -\frac{1}{12} \tag{4.1}$$

for $k = 4$ by:

$$\alpha = \frac{3}{4}; \quad \beta_0 = \frac{3}{8}; \quad \beta_1 = 0; \quad \beta_2 = -\frac{1}{8}; \quad \beta_3 = 0 \tag{4.2}$$

and for $k \geq 5$ by:

$$\alpha = \frac{3}{4}; \quad \beta_i = \frac{1}{k}\left[\frac{1}{4} + \cos\left(\frac{2\pi i}{k}\right) + \frac{1}{2}\cos\left(\frac{4\pi i}{k}\right)\right] \tag{4.3}$$

Using this scheme, the following heuristic is used to calculate the position of the odd vertex: If both vertices of an edge are regular vertices, the weights of the original Butterfly scheme are used (Figure 4.3 (a)). If the edge connects a regular and an irregular vertex, the modified mask (Figure 4.3 (b)) is used to calculate the position by the weighted sum of the irregular vertex and its neighbors. If both vertices adjacent to the new vertex are irregular, Zorin et al. [140] propose to evaluate the new mask for both of them and calculate the position by building the arithmetic average out of the two results. The position of odd boundary vertices is calculated using the 1-dimensional four-point scheme in Figure 4.4.

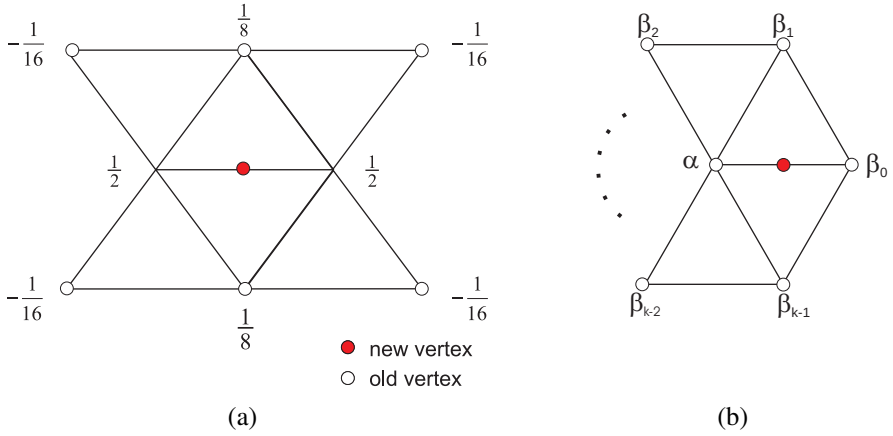


Figure 4.3: Masks of Butterfly scheme. (a) Mask for interior odd vertices with regular neighbors. (b) Mask for interior odd vertices with extraordinary neighbor. k is the valence of the adjacent extraordinary vertex with weight α.

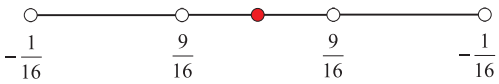
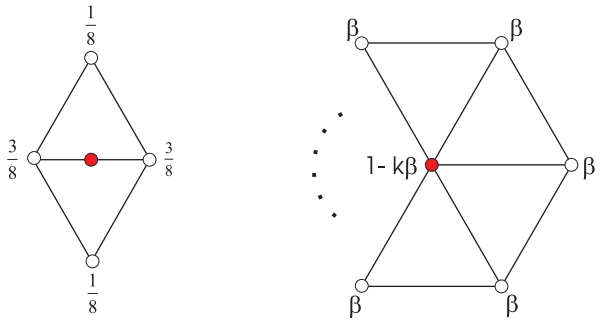


Figure 4.4: Mask for odd boundary vertices.



(a) (b)

Figure 4.5: Edge (a) and vertex mask (b) of the Loop scheme for the calculation of odd and even vertices.

4.2.2 Loop Subdivision

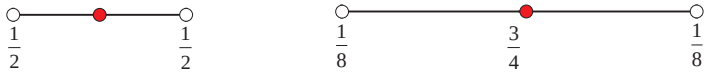
The Loop scheme, proposed by Charles Loop [90], is a simple representative of the approximating subdivision schemes. It is derived from the quartic three-directional box spline, which produces C^2-continuous surfaces on regular meshes. At extraordinary vertices only C^1-continuous surfaces are generated.

Subdivision Rules: Similar to the Butterfly subdivision scheme each face of the mesh is split 1-to-4. However, as the Loop scheme is approximating both odd and even vertices are moved. The new positions of odd and even points are determined by an affine combination of the neighboring vertices with the weights given in the edge mask (Figure 4.5 (a)) and the vertex mask (Figure 4.5 (b)) respectively. The weights in the vertex mask depend on the valence k of the vertex and are

$$\beta = \frac{1}{k}\left[\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{k}\right)^2\right].$$

The new positions of boundary vertices are calculated using special edge and vertex masks (Figure 4.6), which produce cubic spline curves along the boundary. Equally to the Butterfly scheme, the boundary masks only take points along the edge into account.

As Loop's original subdivision scheme only yields C^1-continuity around irregular vertices, Prautzsch et al. [105] proposed modifications of the method that yield C^2-continuity also around irregular vertices. These modifications, however, also had major disadvantages like negative weights in the masks leading to the violation of the convex hull property. Additionally, zero curvature at irregular vertices, and therefore flat areas on the surface resulted. To overcome these limitations, Loop [91] presented an enhancement of his subdivision scheme which



(a) (b)

Figure 4.6: Boundary mask of the Loop scheme: (a) edge mask; (b) vertex mask.

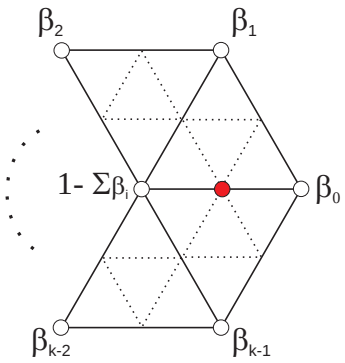


Figure 4.7: Edge mask of the modified Loop scheme.

conserves the convex hull property and results in surfaces with bounded curvature. This is achieved by a new edge mask (Figure 4.7) derived from a cubic polynomial with a larger support than the original edge mask. For vertices with valence $k \geq 6$ the weights β are calculated as follows:

$$\beta_i = M_k\left(\cos\frac{2\pi i}{k}\right).$$

with

$$\begin{aligned} M_k(u) &= a(1+u)(b+u)^2 \\ a &= \frac{2\lambda^3}{k(1-\lambda)} \\ b &= \frac{1}{\lambda} - \frac{3}{2} \\ \lambda &= \frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{k} \end{aligned}$$

The weight for vertices with valence $k < 6$ can be calculated using the following table. Note, for $k = 3$ this corresponds to the original Loop subdivision scheme.

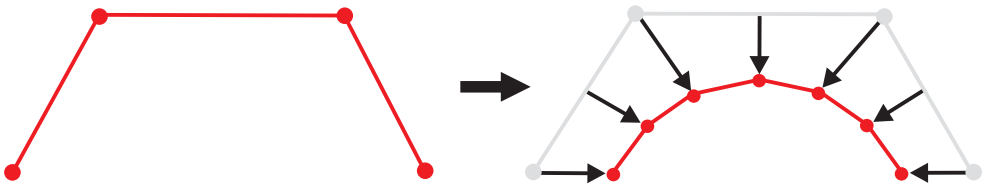


Figure 4.8: The subdivision scheme displaces the vertices of a mesh. Here the effect of an approximating subdivision scheme is shown schematically, hence, all vertices are set to new positions.

k	$M_k(u)$	$1 - \sum \beta_i$
3	$\frac{1}{6}\left(\frac{5}{4} + u\right)$	$\frac{5}{8}$
4	$\frac{1}{2}\left(\frac{1}{2} + \frac{3}{8}u\right)^2$	$\frac{41}{64}$
5	$\frac{3+\sqrt{5}}{32}\left(\frac{5-\sqrt{5}}{5} + u\right)^2$	$\frac{31+5\sqrt{5}}{64}$

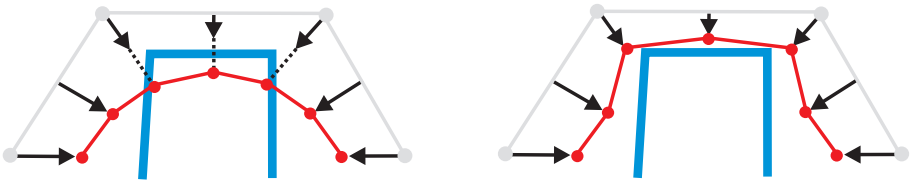
Similar to the modified Butterfly scheme, the edge mask is asymmetric with respect to the incident vertices. To handle this ambiguity, Loop proposes to apply the same heuristic using the arithmetic average as in Section 4.2.1.

4.3 Collision Detection during the Subdivision step

As the subdivisions of a mesh alters the vertex position, collisions and self-collisions between the simulated objects may occur. Especially for meshes close to other objects, as cloth meshes often are, the probability for collisions may be high. In this section we detail how continuous collision detection can be utilized in order to resolve these collisions after each subdivision step.

4.3.1 Collision Detection

To detect the collisions created by the subdivision step we use the two step approach using a k-DOP hierarchy and the continuous collision detection presented in Section 2.4. First, the initial vertex position $\vec{p}_{initial}$ as well as the vertex position in the subdivided mesh $\vec{p}_{subdivided}$ is added to the BVs of the BVH. This assures that any collision between these two positions can be detected. After the BVH reports collision candidates, the continuous collision detection needs the



(a) (b)

Figure 4.9: Collisions are caused by the movement of the vertices in subdivision step (a); all collisions are resolved by repositioning the vertices along the displacement vectors (b).

displacement vector $\vec{s}$ of all involved vertices:

$$\vec{s} = \vec{p}_{subdivided} - \vec{p}_{initial}.$$

For even vertices this is the vector between their position in the control mesh and the new positions calculated by the corresponding vertex mask. Note that for even vertices in interpolating schemes there is no displacement and therefore $\vec{s} = 0$. For odd vertices newly generated in the subdivided mesh, the old position is set to the center of the corresponding edge. The new position is calculated with the edge mask of the subdivision scheme. In Figure 4.8 the displacement vectors are visualized for an approximating subdivision method.

The continuous collision detection then reports the exact point of collision of two primitives (Figure 4.9 (a)) between the initial position and the position of the subdivided mesh. To this point a collision time t is assigned, assuming $t = 0$ for the initial mesh and $t = 1$ for the subdivided mesh. In-between these two states a linear movement is supposed. For vertices that are involved in multiple collisions only the earliest time of collision is kept.

4.3.2 Collision Response

In order to resolve the occurred collisions during subdivision, we start with the finer mesh and sequentially move concerned vertices in direction toward their initial collision-free position just before the point of collision. This new position $\vec{p}_{corrected}$ can be calculated for the vertices as follows:

$$\vec{p}_{corrected} = \vec{p}_{initial} + (\vec{s} \cdot (t - \varepsilon)) \tag{4.4}$$

where a tolerance value ε is considered. On the one hand this value is needed

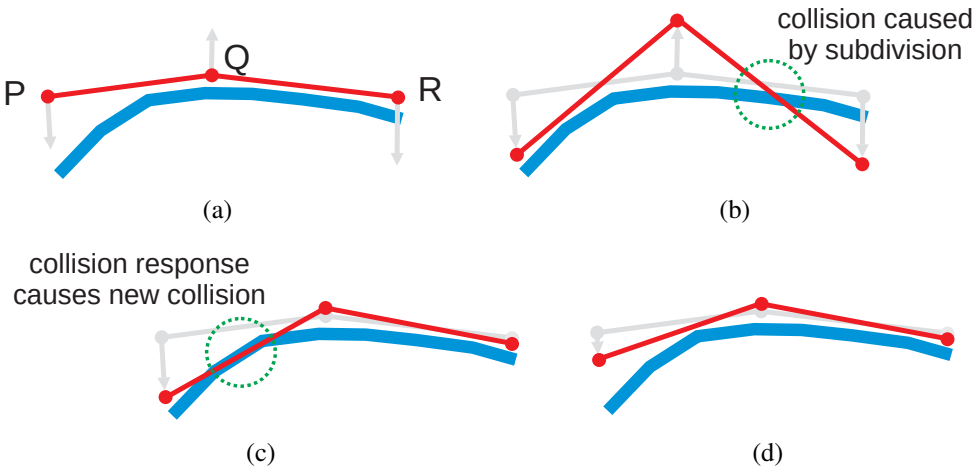


Figure 4.10: In (a) the initial mesh together with the designated displacement vectors of the vertices is shown. The collision detection finds a collision between edge $\overline{QR}$ due to the displacement (b). Hence, Q and R are only moved a small distance. As in the initial collision detection no collisions were found for the edge $\overline{PQ}$ (or for any primitive the vertex P is part of) the vertex P is moved to its designated position (c). This movement now causes a subsequent collision, which is treated by another iteration of the collision detection and response (d).

to account for rounding errors, on the other hand it also allows to model the cloth thickness by assuring a minimal distance between the cloth and other objects.

Since each vertex of a mesh belongs to different edges and faces the collision response at this vertex can cause new collisions of the neighboring elements (Figure 4.10 (a)-(c)). Hence, to resolve all collisions several iterations of the collision detection algorithm with the new positions may be necessary (Figure 4.10 (d)). In all of our simulated examples, however, we found that two iterations are enough to obtain a collision-free mesh. Note that it can be assured to find a collision-free state, as in the worst case, all vertices can be reset to their initial positions before the subdivision step. As the displacements due to the collision response usually remain small, also no artifacts in combination with textures appear.

4.4 Results

Using TüTex, we compared both the visual results and the performance of the two described subdivision methods, namely the modified Butterfly and the modified Loop method. Therefore, different clothing was simulated in a dynamic scene

(a) (b) (c)

Figure 4.11: Comparison between the presented subdivision methods applied to a shirt and the initial mesh (a): two steps of the modified Loop scheme (b); two steps of the modified Butterfly scheme (c).

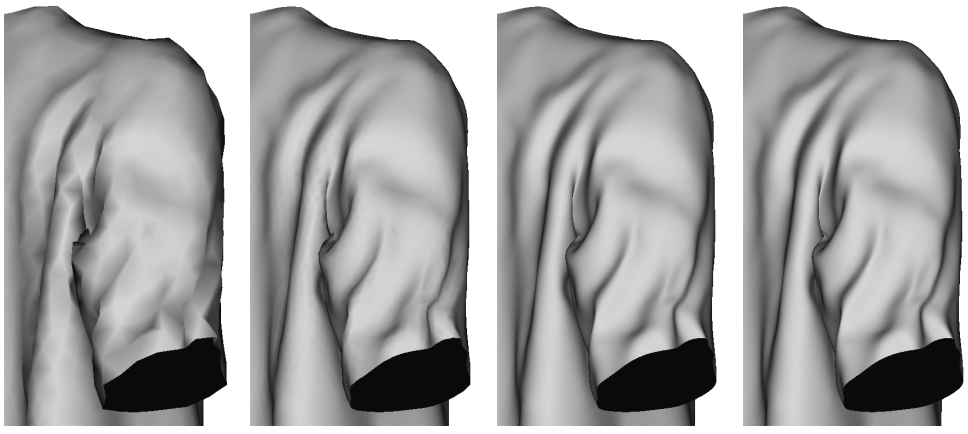
with the TüTex system and multiply subdivided with the presented methods. We used a shirt (3414 vertices and 6712 triangles) and a pair of trousers (3219 vertices and 6362 faces) to dress a walking avatar. Though both meshes have a fairly high resolution, without subdivision it is still possible to see the polygonal nature of the models (Figure 4.11 (a) and 4.12 (a)). Especially in areas with small wrinkles or at the edges this becomes apparent and is disturbing.

In Figure 4.11 (b) and (c) respectively in Figure 4.12 (b) and (c) the garment meshes are shown after two subdivision steps with the modified Loop and the modified Butterfly method. The according data of the subdivided meshes is found in Table 4.2. For both models, shirt and trousers, the differences between interpolating Butterfly method and approximating Loop method are obvious. For the Butterfly method, all vertices of the initial mesh are also part of the subdivided mesh. Therefore sharp irregularities remain in the mesh and are not smoothed out. They are rather intensified for coarse meshes with many wrinkles. The approximating Loop method, however, nicely smoothes out these irregularities and yields a more natural appearance of the subdivided meshes. The implemented subdivision methods rapidly converge to the limit surface as visualized in Figure 4.13 for the shirt subdivided with the modified Loop scheme. After two subdivision iterations almost no further improvements of the smoothness are visible.

To test the collision-free subdivision, we simulated a piece of cloth draped over a table. The coarse cloth mesh with 255 triangles (Figure 4.14 (a)) is sub-

(a) (b) (c)

Figure 4.12: Comparison between the presented subdivision methods applied to a pair of trousers and the initial mesh (a): two steps of the modified Loop scheme (b); two steps of the modified Butterfly scheme (c).



(a) Initial mesh (b) 1 level (c) 2 levels (d) 3 levels

Figure 4.13: Different number of levels of the modified Loop subdivision applied to a shirt.

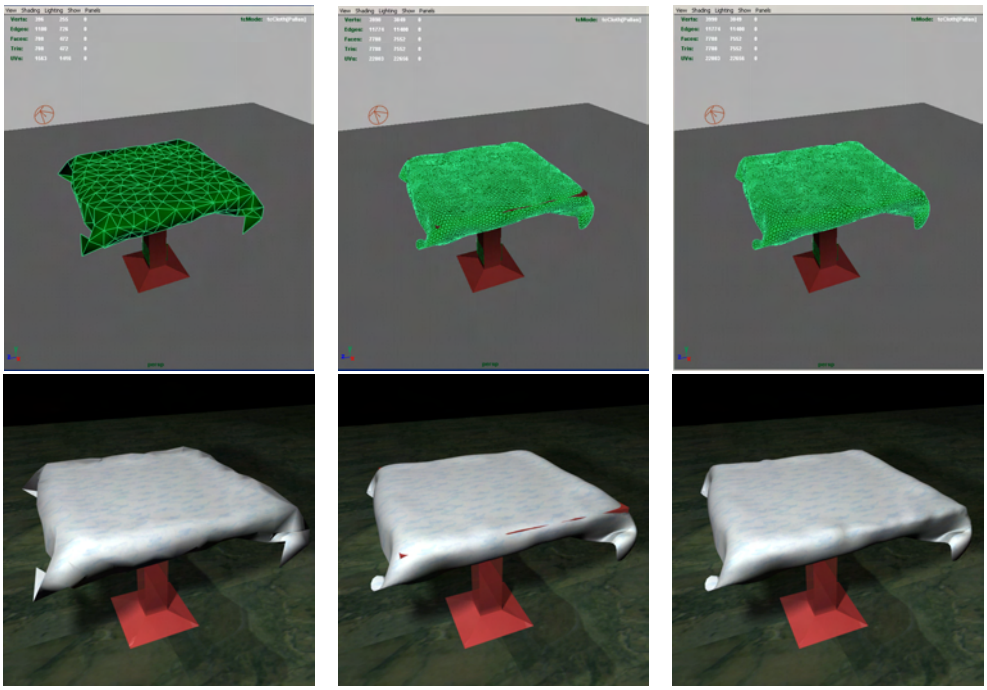
Subdivision steps	0	1	2	3
Shirt mesh				
vertices	3414	13542	53934	215261
irregular vertices	1637	1637	1637	1637
% of irregular vertices	48%	12%	3%	$< 1\%$
faces	6712	26848	107392	429568
Trousers mesh				
vertices	3219	12801	51051	203895
irregular vertices	1480	1480	1480	1480
% of irregular vertices	46%	12%	3%	$< 1\%$
faces	6362	25448	101792	407168

Table 4.2: Data of subdivided meshes after 0, 1, 2 and 3 subdivision steps.

divided twice with the modified Loop scheme. Due to the displacement of the vertices, collisions between table and cloth appear (Figure 4.14 (b)). These collisions are corrected after each subdivision step using the presented approach. In each step, one iteration of the collision detection and response was enough to obtain a collision-free subdivided cloth mesh (Figure 4.14 (c)).

Another test scene closer to the application in a garment simulator is the animation of a walking avatar (28784 triangles) dressed with a shirt and a pair of trousers (Figure 4.15). To achieve fast cloth simulations, low resolution cloth meshes are chosen (shirt 1918 triangles, trousers 1962 triangles). As an example, out of the 60 simulated frames, the frames 4, 48 and 59 are shown in the illustrations. After the physically-based simulation, the frames are subdivided three times by using the modified Loop subdivision (Figure 4.15). The resulting collisions are then corrected using the described approach. For these meshes also one iteration of the detection and response algorithm yields a collision-free mesh. Close-ups of the shoulder region, where the collisions are very apparent, are shown in Figure 4.16.

The mesh data and the corresponding performance measurements are shown in Table 4.3. Though only a small number of vertices collide, this yields noticeable artifacts. The maximum displacement in the vertex position reduces with more and more subdivision steps. Hence, the number of collisions also gets smaller. In contrast, the run-time of the first step of the collision detection, the collision detection using k-DOPs, strongly increases from one subdivision step to the next with increasing number of triangles. The run-time of the second step of the collision detection, the continuous collision detection, stays almost constant with the reducing maximum displacement of the vertices. Hence, in this example the num-



(a) (b) (c)

Figure 4.14: Table cloth simulated using collision-free subdivision. The table cloth is first simulated using a coarse mesh (a). The subsequent Loop subdivision causes collisions between the cloth and the table (b). These collisions are corrected using the continuous collision detection and the presented collision resolution (c).

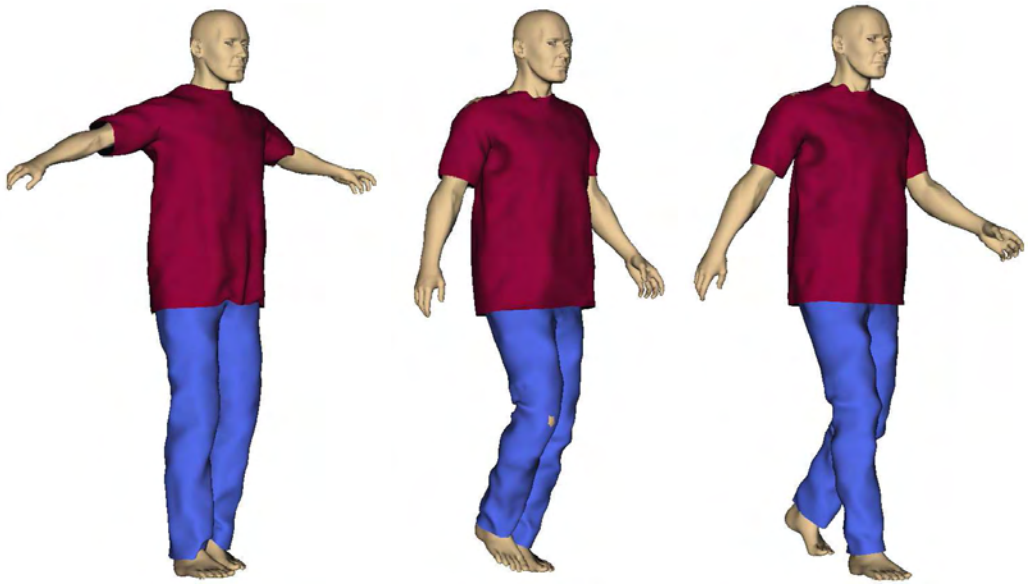
ber of triangle pairs within the maximum displacement distance stays constant leading to the nearly constant computation time of the continuous collision detection. All the run-time measurements were performed on a PC with AMD Athlon 2800+ processor and 1024 MB DDR-SDRAM main memory.

4.5 Summary

In order to obtain visually appealing simulation results with smooth cloth surfaces high resolution meshes are required. They either can be obtained by simulating them directly or by simulating coarse meshes that are post-processed after the simulation. Since cloth simulation with high resolution meshes is very time consuming, the latter is often preferred, especially since post-processing is not necessary for each time step but only for frames that are rendered. In this chapter, we therefore presented an approach for the post-processing of simulated cloth

	frame 4		frame 48		frame 59	
	shirt	trous.	shirt	trous.	shirt	trous.
subdivision step 1						
vertices	3910	3965	3910	3965	3910	3965
faces	7672	7845	7672	7845	7672	7845
maximum displacement [mm]	12.7	10.9	12.5	14.9	12.9	12.4
close faces	10177	20349	11037	18146	10148	17668
corrected vertices	12	2	45	14	20	9
continuous collision det. [ms]	336	676	366	603	345	588
edge-edge tests [ms]	10	16	12	15	10	16
vertex-triangle tests [ms]	3	4	4	3	3	4
k-DOP collision det. [ms]	572	638	558	934	766	766
subdivision step 2						
vertices	15494	15779	15494	15779	15494	15779
faces	30668	31392	30668	31392	30668	31392
maximum displacement [mm]	2.9	3.2	3.6	4.1	3.5	3.5
close faces	9801	13479	12783	12535	11885	12161
corrected vertices	9	0	20	5	10	2
continuous collision det. [ms]	320	442	418	410	390	399
edge-edge tests [ms]	3	4	5	4	4	4
vertex-triangle tests [ms]	1	1	1	1	1	1
k-DOP collision det. [ms]	880	918	972	954	1213	1310
subdivision step 3						
vertices	61687	62951	61687	62951	61687	62951
faces	122752	125568	122752	125568	122752	125568
maximum displacement [mm]	1.4	1.4	1.9	1.4	1.5	1.5
close faces	16742	15970	24561	18008	21219	18317
corrected vertices	0	1	5	0	2	1
continuous collision det. [ms]	543	520	798	586	705	597
edge-edge tests [ms]	2	2	3	2	2	3
vertex-triangle tests [ms]	1	1	1	1	1	1
k-DOP collision det. [ms]	5261	6226	5874	6411	5054	5404
total						
for 1 step [s]	0.92	1.33	0.94	1.56	1.12	1.38
for 2 steps [s]	2.13	2.70	2.34	2.92	2.73	3.10
for 3 steps [s]	7.93	9.45	9.01	9.92	8.49	9.10

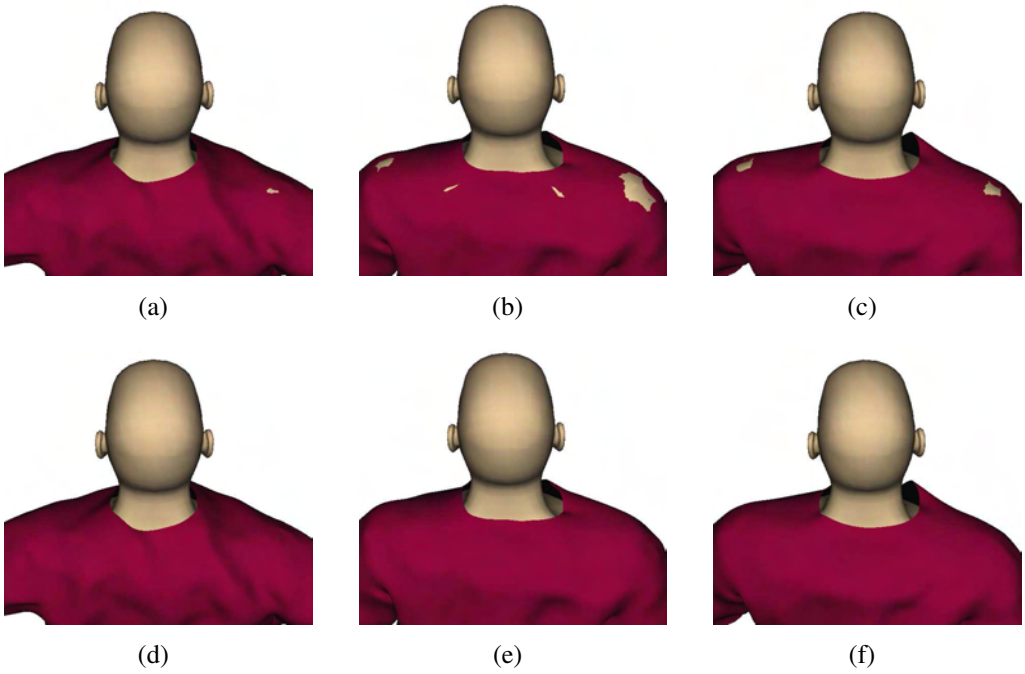
Table 4.3: Mesh data and run-time of collision detection for three subdivision steps at the different animation frames.



(a) (b) (c)

Figure 4.15: Animation of walking avatar with three subsequent steps of the modified Loop subdivision causing collisions. The following three frames are analyzed in detail: frame 4 (a); frame 48 (b); frame 59 (c)

meshes using different subdivision methods. We are the first who compared approximating and interpolating subdivision schemes represented by the modified Loop and the modified Butterfly scheme with respect to their practicability in this context. Experimentally, we were able to show that the approximating Loop scheme yields the nicest results when subdividing low resolution cloth meshes. If subdivision schemes are applied to the cloth meshes, old and new vertices are moved and therefore collisions may be caused. We resolve these collisions after detecting them using a bounding volume hierarchy in combination with a continuous collision detection as detailed in Chapter 2. The presented methods for collision-free subdivision can also be applied in combination with automatically generated wrinkle maps as shown in the following chapter.



(a) (b) (c)

(d) (e) (f)

Figure 4.16: Close-up of animated shirt subdivided with the modified Loop scheme in Figure 4.15. The frames 4, 48 and 59 are shown without collision detection and response (a)-(c) and with collision detection and response (d)-(f).

CHAPTER 5

Texture-based Wrinkle Simulation

5.1 Introduction and Related Work

For virtual clothing, wrinkles are the essential part of the simulation and make them look realistic. The occurrence of wrinkles in textiles is a complex phenomenon caused by the interaction of inner material forces and friction forces. It is very challenging to model them correctly even for high resolution meshes. As the simulation of high resolution meshes still yields very long simulation times, in the last chapter we proposed the collision-free subdivision method to post-process coarse meshes. This approach, however, only smoothes the virtual textiles and does not add fine details like realistic folds and wrinkles. To achieve similar good visual results using low resolution meshes as can be obtained by simulating high resolution meshes we propose a post-processing method that generates wrinkle textures based on the deformation of the underlying triangles. These textures are used to visualize fine wrinkles on the cloth and are combined with the collision-free subdivision. This yields smooth, high resolution meshes with realistic folds and wrinkles.

Due to the inner material forces, tissue rather tends to buckle than to compress. For the post-processing method presented here, we therefore assume that

cloth conserves its area and calculate the height of wrinkles that would result for triangle deformed in-plane. This height field of a triangle is generated based on procedurally generated textures with different orientations and stored as wrinkle texture. By interpolating these wrinkle textures over the complete cloth mesh a bump or displacement map is obtained.

In previous work on geometric wrinkle generation by Hadap et al. and Volino et al. [62, 130] the use of modulated wrinkle textures has already been proposed in order to enhance the simulation of coarse cloth meshes. The main drawback of their method, however, is the need for much user interaction in the creation of the underlying wrinkle textures. Further methods have been proposed for geometric wrinkle generation for human skin [86, 39], whereas other publications focus on the correct buckling behavior of cloth to model physical reality [36, 37, 34]. For their methods, however, high resolution meshes are necessary to obtain visually good results. All methods creating wrinkles geometrically do not resolve collisions that are caused by the added wrinkles.

In this chapter we extend the ideas of Hadap et al. without the necessity of user interaction and address the problem of occurring collisions. First, we describe how wrinkles can be calculated from the inner strain of the mesh. Contrary to earlier, purely geometric approaches, this method is based on physical principles (Section 5.2) as we reuse the deformation tensor that can easily be calculated during the finite element cloth simulation [51] in TüTex. In Section 5.3 we describe how the wrinkle textures can be generated procedurally, which are then applied to the mesh by bump or displacement mapping (Section 5.4). Since for the latter method new collisions for the simulated meshes may occur we use the adaptive subdivision algorithm coupled with continuous collision detection as described in Section 4.3. For the displacement mapping, two software implementations are proposed. Either the subdivision methods presented in the previous chapter may be used, or an alternative approach suggested by Moule and McCool [98] is applied. To show the validity of the presented approach, in Section 5.5 we discuss our results. Several pieces of cloth simulated with TüTex and post-processed with the described methods are presented. Furthermore, we compare the quality and runtime of high resolution cloth simulations without added wrinkle textures to cloth simulations based on coarse meshes and subsequent post-processing as proposed in this chapter (Section 5.5). The results presented in this chapter were published in [6].

5.2 Wrinkle Coefficients from Strain

Regarding the inner material force, textiles have very high compression stiffness and relatively low bending and shearing stiffness which results in immediate buckling. If we use a coarse mesh to represent a tissue, this buckling effect cannot be modeled correctly due to lack of degrees of freedom. Additionally, if we use the corresponding material parameters for the coarse mesh this would result in very high compression forces. Hence, we use slightly lower material parameters for compression stiffness and couple the deformation of the mesh to the generation of wrinkle textures. With this model, we calculate the rough movement of the cloth by the simulation of the mesh whereas fine detail is added by textures depending on the inner deformation state of the mesh. For deformable objects several deformation measures have been introduced to determine the inner strain of the object. Let φ be the configuration mapping of a parameterized surface with respect to the local coordinate system (u, v), then $J = (\frac{\partial \varphi}{\partial u}, \frac{\partial \varphi}{\partial v})$ is called the deformation gradient. The displacement δ is defined as $\delta = \varphi - Id$, where Id is the identity matrix. The most common deformation measures are the symmetric Green's strain tensor

$$\epsilon^G = \frac{1}{2}(JJ^t - Id)$$

and its linear approximation, the Cauchy strain tensor ϵ^C. The Cauchy strain tensor, however, is not rotational invariant [29]. Etzmuß et al. [51] introduced a rotationally invariant linear deformation tensor due to corotational finite elements. By this approach, the deformation gradient can be locally decomposed into a rotation R and a pure deformation U, i.e. $J = R \cdot U$ and Green's tensor can be written as

$$\epsilon^G = \frac{1}{2}(U^2 - Id) \; .$$

Due to the large in-plane tension stiffness of cloth which creates wrinkles we couple our wrinkle texture to the deformation U of the simulated mesh. Pure rotations have no influence on the deformation and the formation of wrinkles. For every triangle in the simulated mesh we obtain a pure deformation transformation as

$$U : \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ b & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{5.1}$$

with respect to the local coordinate systems of the triangle with the inverse

$$U^{-1} : \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a' & b' \\ b' & d' \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix} \; .$$

Additionally to this deformation measure, we briefly present the measure for the deformation of the underlying material that has been introduced by Hadap et al. [62]. Usually in simulation modules for textiles, an initial plain mesh, i.e. garment pattern, is given to determine the rest state of the textile. Each triangle of the simulated, deformed mesh can then be described by a deformation transformation with respect to a local two-dimensional in-plane coordinate system, where the axes of each local coordinate system are chosen such that the x-axis, respectively x'-axis is given by one edge of the triangle and the y-, respectively y'-axis, is chosen such that we obtain a right-handed coordinate system which lies in the plane of the triangle. This results in the following deformation transformation

$$\tilde{U} : \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \tilde{a} & 0 \\ \tilde{b} & \tilde{c} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{5.2}$$

with the corresponding inverse.

We want to point out that this measure is easy to implement for an existing cloth simulation module but gives only a heuristic measure for the deformation of the underlying textile. Since in our simulation module [4, 51] we get the deformation gradient from the finite-element calculations, we use it to obtain the wrinkle textures as well.

To map wrinkles on the underlying triangles of our simulated mesh we start with a continuous differentiable wrinkle function

$$f : \Delta \in \mathbb{R}^2 \rightarrow \mathbb{R}$$

for the initial mesh, respectively for an initial triangle Δ. Hence, we can map the initial wrinkle function on the deformed triangle $\Delta' = U(\Delta)$ by

$$f'(x', y') : \Delta' \rightarrow \mathbb{R} : (x', y') \mapsto f(U^{-1}(x', y')).$$

The area of the undeformed triangle Δ is given by

$$A(\Delta, f(x, y)) \;=\; \int\limits_{\Delta'} \sqrt{1 + \left(\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right)} dx dy. \tag{5.3}$$

Since we use the deformation of the underlying triangle of the simulated mesh to determine the wrinkling behavior, we have to conserve the area of the wrinkle function f over the triangle. In our case, this is done by scaling the function f by the *modulation factor* h.

By a series expansion of the parameterized surface area A given by

$$
\begin{aligned}
A(\Delta', hf'(x', y')) &= \int_{\Delta'} \sqrt{1 + h^2 \left(\left(\frac{\partial f'}{\partial x'} \right)^2 + \left(\frac{\partial f'}{\partial y'} \right)^2 \right)} dx' dy' \\
&= \int_{\Delta} \frac{1}{a'd' - b'^2} \left(1 + h^2 \left(a' \frac{\partial f}{\partial x} + b' \frac{\partial f}{\partial y} \right)^2 \right. \\
&\quad \left. + h^2 \left(b' \frac{\partial f}{\partial x} + d' \frac{\partial f}{\partial x} \right)^2 \right)^{1/2} dxdy \\
&=: \int_{\Delta} I(h, a', b', d')\, dxdy \qquad (5.4)
\end{aligned}
$$

we obtain as linear approximation

$$
\begin{aligned}
A(\Delta', hf'(x', y')) &= \int_{\Delta} (I(1, 1, 0, 1) + \frac{\partial I}{\partial h} \cdot (h - 1) \\
&\quad + \frac{\partial I}{\partial a'} \cdot (a' - 1) + \frac{\partial I}{\partial b'} \cdot b' \\
&\quad + \frac{\partial I}{\partial d'} \cdot (d' - 1))\, dxdy\,. \qquad (5.5)
\end{aligned}
$$

The area conservation then yields

$$
A(\Delta, f(x, y)) = A(\Delta', hf'(x', y'))\,. \qquad (5.6)
$$

By inserting Equation (5.5) into Equation (5.6), we get

$$
\begin{aligned}
A(h, \Delta, f(x, y)) = C_0 &+ C_1(h - 1) + C_2 a' \\
&+ C_3(b' - 1) + C_4(d' - 1) \qquad (5.7)
\end{aligned}
$$

with the coefficients

$$
\begin{aligned}
C_0 &= \int_{\Delta} I(1,1,0,1)\, dxdy \\
C_1 &= \int_{\Delta} \frac{\partial I}{\partial h}\, dxdy\,, \quad C_2 = \int_{\Delta} \frac{\partial I}{\partial a'}\, dxdy\,, \\
C_3 &= \int_{\Delta} \frac{\partial I}{\partial b'}\, dxdy\,, \quad C_4 = \int_{\Delta} \frac{\partial I}{\partial d'}\, dxdy\,.
\end{aligned}
$$

The coefficient C_0 corresponds to $A(\Delta, f(x,y))$, thus, the modulation factor h is calculated as

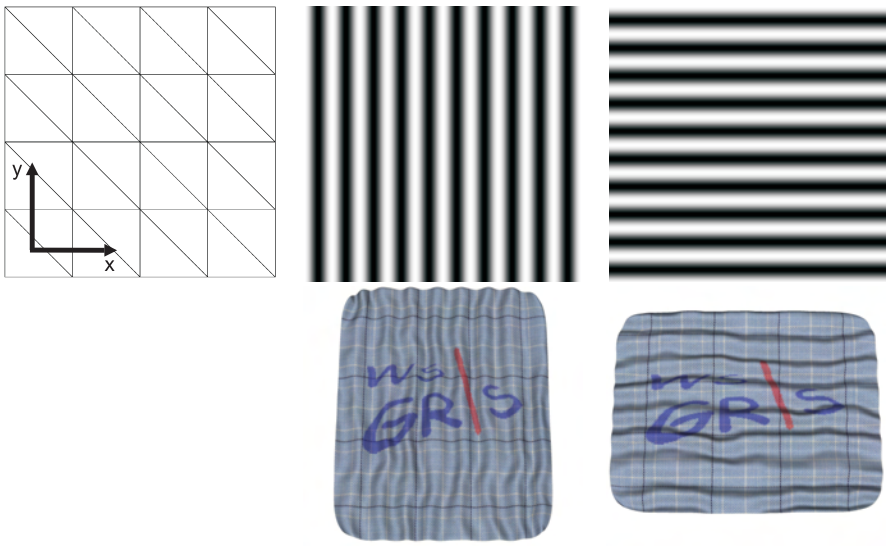
$$
h = \begin{cases} 1 & \text{for } C_1 = 0, \\ 1 - \frac{1}{C_1}(C_2(a'-1) + C_3 b' + C_4(d'-1)) & \text{else.} \end{cases} \tag{5.8}
$$

The modulation factor for the alternative deformation transformation in Equation (5.2) is calculated analogously.

While the wrinkle coefficients C_1 to C_4 are computationally expensive but can be calculated in a pre-computation step, the modulation factor h is easily computed for every triangle during the simulation. Analyzing this algorithm it turns out that the modulation factor favors special wrinkle functions f, where the wrinkling is orthogonal to the strain. Hence, the modulation factor is high if the wrinkle pattern fits the underlying deformation well. This leads us to the idea of multilayered textures in the next section. Since we obtain one modulation factor per triangle we linearly interpolate the factors across the triangles to end up with a smooth modulation map. Note that by our linear approximation of the area function and the linear smoothing of the scaling, area conservation is no longer preserved exactly.

5.3 Multilayered Textures

As real wrinkles do not appear only in one direction within garments, the use of multiple textures is necessary. Already Hadap et al. [62] presented the concept of multifold wrinkling, i.e. the combination of different wrinkle textures was presented but they use two user defined wrinkle patterns for the garment which have to be orthogonal and require much interaction and knowledge of the user. Here, we extend the idea of switching between wrinkle patterns by generating



(a) (b)

Figure 5.1: Multilayered wrinkle textures with visualization of the simulated piece of cloth ($\omega = 10$, $n = 2$).

multilayered wrinkle patterns procedurally and blending them according to the deformation of the mesh.

The multilayered textures are built using a simple sine-function, where the intensity of the ith texture is defined by

$$\begin{aligned} l_i(x,y) &= \frac{1}{2}(\sin(2\pi \cdot \omega \cdot x_{i,n}) + 1) \,, \\ x_{i,n} &= x \cdot \cos\left(\frac{\pi \cdot i}{n}\right) - y \cdot \sin\left(\frac{\pi \cdot i}{n}\right) \,, \end{aligned}$$

where ω is the parameter defining the number of folds per texture and n the number of textures we want to provide per triangle (Figure 5.2).

To enhance the effect of natural-looking folds, we add Perlin noise [102] to the wrinkle textures (Figure 5.3). Persistency and variance v should be kept low in order to avoid too many high frequencies in the textures. The noise function $noise(x, y)$ is steady and returns values between -1 and 1.

$$l_i(x,y) = \frac{1}{2}(\sin(2\pi \cdot \omega \cdot x_r + \text{noise}(x,y) \cdot v) + 1) \,.$$

Though the textures are very simple, by a combination of them we obtain a large variety of different naturally looking folds. To determine the choice of

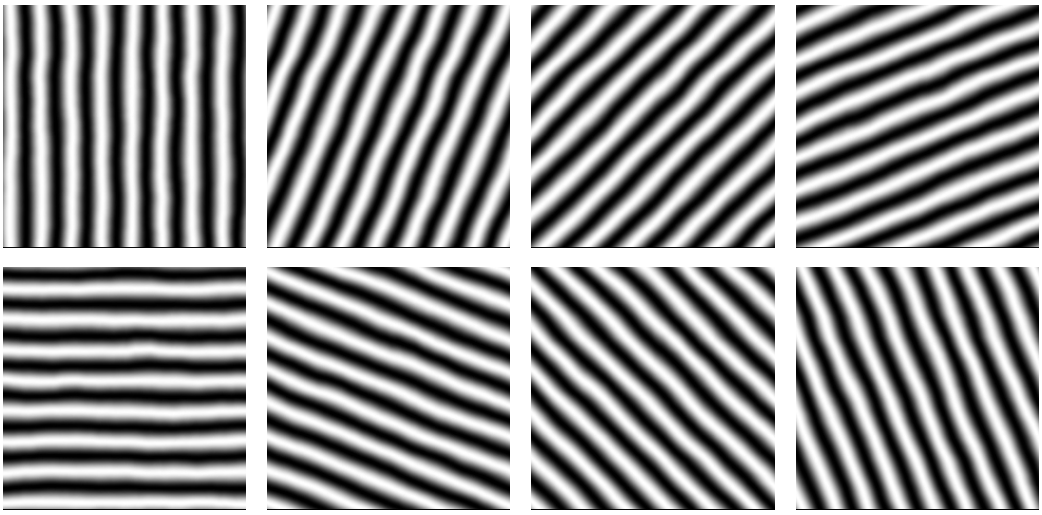


Figure 5.2: Procedural wrinkle textures in eight directions ($\omega = 10$, $n = 8$).

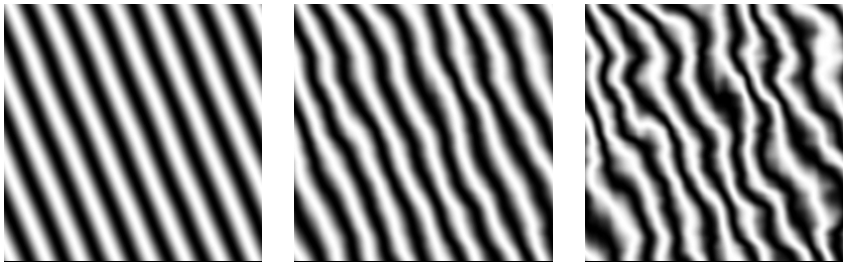


Figure 5.3: Procedural wrinkle textures with different Perlin noise variations ($v = 0, 4, 8$).

wrinkle textures the modulation factors are used as they are very sensitive to the orientation of the wrinkles. If, for example, we apply the wrinkle textures Figure 5.1 (a) and (b) on the quadratic mesh, then a deformation of the mesh in the x-direction results in a higher modulation factor of the texture (a) whereas a deformation in the y-axis leads to a larger factor of texture (b). Hence, in the simulation we employ a linear combination of different wrinkle functions l_i depending on the modulation factor h.

The use of a combination of too many different wrinkles textures to model the wrinkles belonging to a specific deformation, results in high frequencies of the wrinkles. For this reason we restrict ourselves to a combination of at most two wrinkle textures per triangle. Therefore, we first scale all modulation factors belonging to different wrinkle textures per triangle to the interval $[0, 1]$. Then, we choose the textures with the two highest modulation factors. Between these

textures the user may define a transition, hence, only in cases where both modulation factors are almost equal, a linear combination of the textures is calculated. Outside of this zone, only one texture per triangle is considered.

5.4 Wrinkle Texture Rendering

After generating the strain based wrinkle textures we use them in order to enhance and visualize the simulated coarse meshes. Therefore, the textures are applied to the meshes with two commonly used methods. First we implemented a bump mapping technique, which is ideal for a fast visualization during the rendering process. Second, we realized the more intricate method of displacement mapping.

5.4.1 Bump Mapping

Bump mapping was originally presented by Blinn [28]. Instead of modifying the surface directly, the normals are altered corresponding to the given texture and the texture coordinates. Then for each point of the surface the illumination is calculated depending on these normals. This technique is simple, implemented on most current graphic cards and is especially suited for small perturbations of the surface. Bump mapping allows to obtain visual effects of folds, wrinkles and other surface structures and has therefore been used in the simulation of facial folds [136] as well as in cloth wrinkles [62]. The main drawback of bump mapping is that the vertices are not moved and therefore the surface is not deformed. This becomes apparent when the camera is close to the object or the silhouette is regarded.

5.4.2 Displacement Mapping

Displacement mapping was first introduced by Cook [40] as a technique for adding detail to surfaces. Contrary to bump mapping, where only the normals of rendered surface points are altered, displacement mapping modifies also the geometry of the underlying mesh. Hence, also the silhouettes look realistic even for close-ups. Before discussing the problem of collisions and self-collisions caused by displacement mapping, we will give a short overview over this technique and how we adapted it for our purposes.

In displacement mapping a texture is used in order to add its offset coefficients $d(u, v)$ to the vertices $\vec{p}(u, v)$ of the underlying parameterized mesh in normal

direction $\vec{n}_p$. The new vertex position $\vec{p}_{new}$ is calculated as follows

$$\vec{p}_{new}(u,v) = \vec{p}(u,v) + d(u,v) \cdot \vec{n}_p.$$

To show fine details, for this approach a starting mesh with high resolution is needed. However, this contradicts the requisites for our wrinkle simulation: Since high resolution meshes yield high computation times for the physical simulation, we use coarse meshes. To obtain high resolution meshes in spite of this requirement the simulated mesh may be refined after the simulation using subdivision methods. Several software implemented subdivision schemes that are suitable for this purpose have been presented in Chapter 4. Recently algorithms for the subdivision and displacement mapping process were developed, which can also be realized in hardware [44, 68].

Because software implementations are more flexible and current graphic cards still do not support displacement mapping in a comfortable way, we apply the algorithm of Moule and McCool [98], which has also be shown to deliver real time displacement mapping. We use their adaptive subdivision method together with the obtained wrinkle textures to get a wrinkled mesh with finer details. In this method, the triangles are subdivided until they meet a certain criterion. To decide if we have to proceed with the subdivision or to stop, we evaluate an *oracle*. It analyzes the displacement map in the region spanned by the edges of a triangle (Figure 5.4) and yields the maximum and minimum displacement d_{max}, respectively d_{min} of the displacement map in this area. If the difference $d_{\text{max}} - d_{\text{min}}$ is smaller than a user defined threshold we stop the subdivision. Otherwise depending on the result of the oracle for the different edges we apply further refinement steps according to Figure 5.5.

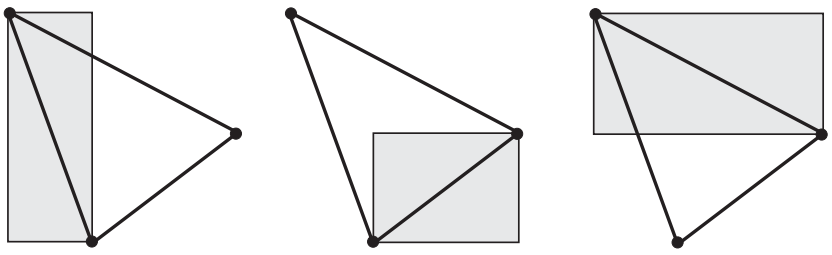


Figure 5.4: Oracle to decide on further subdivision steps: The displacement map is evaluated in the regions spanned by the triangle edges. If the difference between maximum and minimum displacement is smaller than a specified threshold the subdivision is stopped for the corresponding edge. This yields seven different schemes for the triangle subdivision (Figure 5.5)

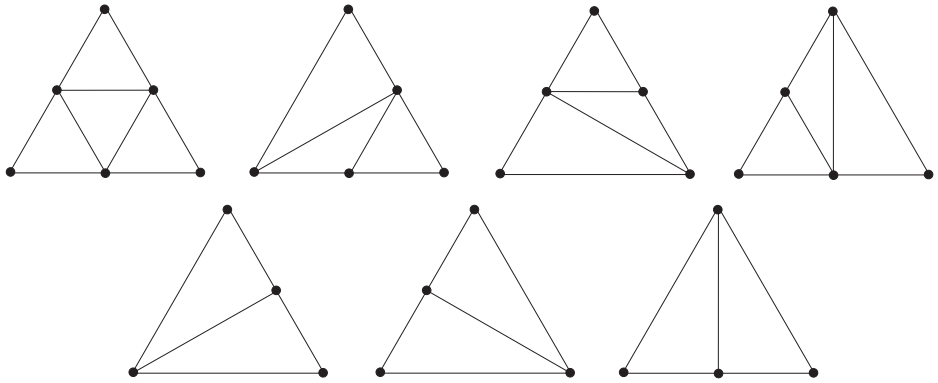


Figure 5.5: After using the oracle in Figure 5.4 to decide which edge needs to be further subdivided, these schemes are used to refine the triangles.

Additionally to the method of Moule and McCool, we are able to use the subdivision methods presented in Chapter 4 to get a high resolution mesh. These methods, however, are view point independent and not adaptive. On the other side, the subdivision methods yield a smoothing of the mesh increasing the visual quality even in areas without displacement.

Due to the displacement mapping and the resulting change of the vertex positions, collisions between the cloth mesh with other objects or with itself may be caused. In previous approaches this was completely ignored, leading to unpleasing visual artifacts. To detect these collisions we propose to use a technique similar to the method presented in Section 4.3 for subdivision. First, the point of collision is detected for each displaced vertex. Then the colliding vertices are set to points just before the collisions occur. Using this new approach, combining collision-free subdivision and strain-based wrinkles textures, the problem of induced collisions is corrected and visually pleasing results are obtained.

5.5 Results

To show the validity of the presented approach, first, the correct choice of the wrinkle textures by our algorithm described in Section 5.2 and 5.3 is visualized by applying strain forces in different directions on a mesh consisting of 32 triangles. The results obtained using eight wrinkle textures are shown in Figure 5.1 and 5.6, where we note realistic wrinkles corresponding to the deformation of the underlying mesh.

For the generated bump or displacement maps of the wrinkle texture we used

Figure 5.6: Pieces of cloth deformed in different directions with added wrinkle textures.

a resolution of 256x256 pixels. For our purposes, higher resolutions do not improve considerably the quality of the simulation while leading to higher calculation times. Whereas very few wrinkle textures do not model the deformation convincingly on the mesh, much initialization time to calculate the wrinkle coefficients is needed if many wrinkle textures are used. Our simulations have shown that already six wrinkle textures yield satisfying results and more than eight textures do not visibly improve the simulation (Table 5.1). We compare the meshes post-processed with our method to the plain subdivided mesh for a shirt and a skirt in Figure 5.7 and 5.8. In all given examples the post-processing using strain-based wrinkles textures increases the realism of the visualization by adding fine wrinkles (Figure 5.7 (c)-(d) and 5.8 (d)-(f)). The clothes without the added wrinkle textures are only smoothed using the collision-free subdivision.

In the final example we combine our results of Chapter 4 and 5 by comparing a coarse mesh simulation of a pair of trousers (1962 triangles) with subsequent wrinkle animation to a simulation of a high resolution mesh (6212 triangles) to evaluate the quality and the calculation times. Though we have no physically correct folds in the low resolution case we obtain visually pleasing and physically plausible results (Figure 5.9). The result obtained by the wrinkle animation of a coarse mesh is of the same visual quality as the one simulated using a high resolution mesh with TüTex. The total computation time, however, is less than a third (Table 5.2). Additionally, the collision-free wrinkle animation is only necessary for frames that are rendered and not for every time step. All calculations presented in this section were done on a PC with AMD Athlon 2800+ processor and 1024 MB DDR-SDRAM main memory.

scene	trousers	shirt	skirt
vertices	1001	917	508
faces	1962	1764	969
wrinkle texture generation			
wrinkle texture	6	6	8
wrinkles per texture	18	18	12
persistency	0.3	0.4	0.3
variation	6.0	6.0	5.0
total wrinkle generation time [s]	2.1	2.1	2.8
initialization			
wrinkle coefficients [s]	5.91	6.29	11.07
total initialization [s]	5.93	6.31	11.09
wrinkle simulation per frame [ms]			
modulation per face	1.8	1.5	1.2
modulation per vertex	1.3	1.2	1.0
generation of wrinkle textures	19.6	19.3	26.6
generation of displacement map	10.7	10.7	14.5
total calculation time per frame	35.7	33.6	44.0

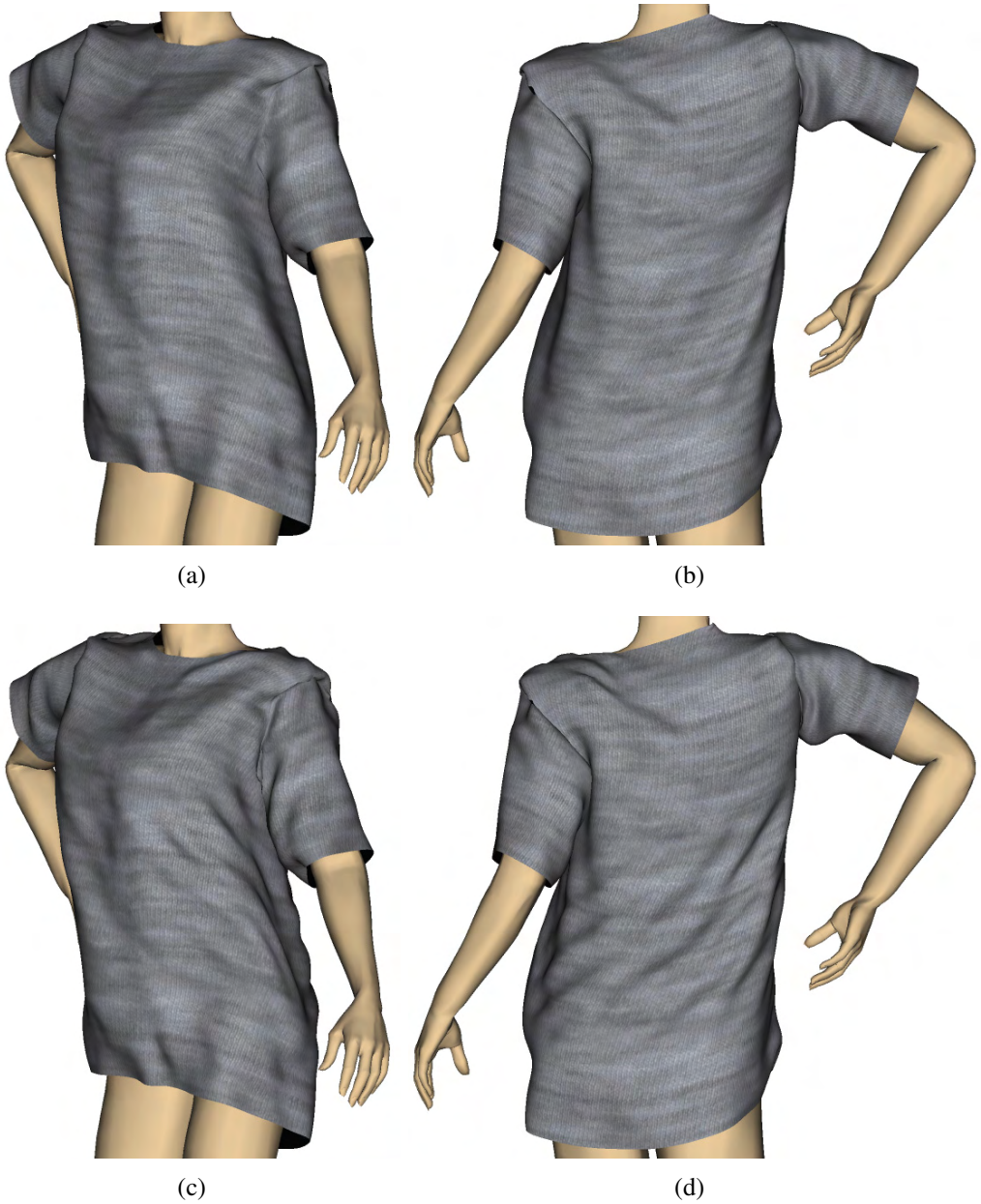
Table 5.1: Calculation times and data for wrinkle animation of trousers, shirt (Figure 5.7) and skirt (Figure 5.8).

	low resolution	high resolution
faces	1962	6212
faces after subdivision	31392	-
average calculation time per simulated second [min.]		
simulation	2.33	12.20
collision free subdivision	1.4	-
wrinkle simulation	0.01	-
total calculation time	3.48	12.20

Table 5.2: Calculation times for the pair of trousers using a low resolution meshes with subsequent wrinkle simulation and for the same pair of trousers directly simulated using a high resolution mesh (Figure 5.9).

5.6 Summary

Wrinkles are the essential part of the simulation of virtual garment and make it look realistic. As in the post-processing of coarse meshes a pure subdivision only yields a smoothing, in this chapter, we have presented a new strain-based method to simulate wrinkles for cloth simulation. Our main contribution is that the underlying wrinkle textures are generated procedurally without the need for user interaction. Furthermore, we are the first who directly reuse the calculated strain tensor of the finite element simulation in order to determine the wrinkle textures. In addition, this approach combines wrinkle textures with collision-free subdivision step in order to realize displacement mapping. Thereby, we achieved very fine details in the final mesh while simulating a rather coarse mesh. Since the method is physically-based the results are visually similar to the simulation of a high resolution mesh without post-processing but can be computed in a significant shorter simulation time.



(a) (b)

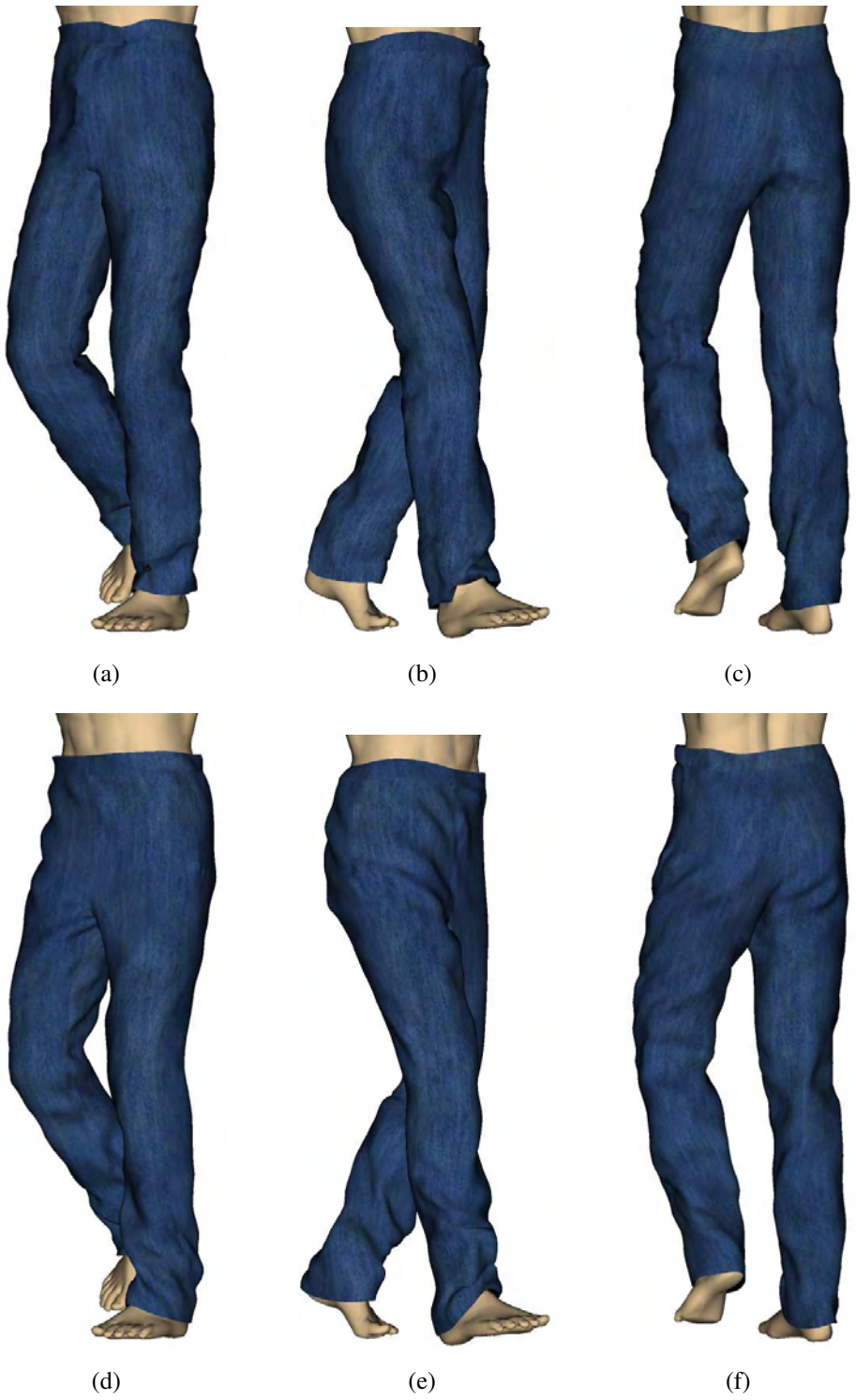
(c) (d)

Figure 5.7: Simulated shirt during a walk animation: Without wrinkle simulation (a)-(b) and with wrinkle animation combined with collision-free subdivision (c)-(d).

(a) (b) (c)

(d) (e) (f)

Figure 5.8: Simulated skirt during a walk animation: Without wrinkle simulation (a)-(c) and with wrinkle animation combined with collision-free subdivision (d)-(f).



(a) (b) (c)

(d) (e) (f)

Figure 5.9: A pair of trousers simulated using a high resolution mesh (a)-(c), and the same pair of trousers simulated using a low resolution mesh with subsequent wrinkle simulation and collision-free subdivision (d)-(f).

CHAPTER 6

Integration into TüTex

6.1 Introduction and Related Work

Since collision detection, collision response and post-processing are an essential part of any cloth simulation system, all the techniques presented in the previous chapters of this thesis were implemented in the cloth simulation system TüTex. This system was developed over the past years [1, 12, 4, 13] at the computer graphics group WSI/GRIS at the University of Tübingen. TüTex is a further development of work done at WSI/GRIS since almost a decade [48, 47].

In addition, due to the increasing power of modern computers the physically correct simulation of cloth became interesting for a wider range of applications. Hence, commercial animation packages like Maya by Alias [19] or XSI by Softimage [117] nowadays not only include modules for the simulation of hair and fluids, but also modules for the simulation of cloth. The applications for these animation packages are usually restricted to entertainment and illustration purposes. Only recently the more conservative textile industry was attracted by the simulation of textiles, since many appealing applications exist in their domain. These range from the virtual try-on of garments on avatars of real people [41, 13, 14, 7] to rapid prototyping and systems supporting the design process of clothing [78].

The idea of virtual try-on systems is to support the customer's decision making by a realistic simulation and visualization of virtual garments, before the real garments are actually manufactured or, in the case of internet shopping, sent to the customer. This yields reduced stock keeping or shipping costs for clothing retailers and also can support automatic manufacturing of made-to-measure garments based on the customer's personal preferences and needs. In this context, TüTex became an integral part of the research project Virtual Try-On, in which textile industry, textile research institutes and computer graphics groups worked together in order to realize these visions.

In the following, we first discuss how the collision detection and post-processing methods presented in this thesis are integrated into the TüTex cloth simulation system (Section 6.2). In Section 6.3 we present the objectives of the Virtual Try-On project and detail which of them were realized during the project. The integration of the TüTex system into the 3D modeling software Maya by Alias is described in Section 6.4.

6.2 TüTex

All the techniques previously described in this thesis have been incorporated into the cloth simulation software *TüTex*. An overview of all the components is shown in Figure 6.1. It mainly consists of the underlying physical model, algorithms for numerical time integration, the presented collision detection and response methods, and the post-processing unit. Additionally, interfaces for the input of avatars, cloth models, and material parameters are provided as well as for rendering and user interface components. The complete system was implemented in C++, and the object oriented architecture helped to support the design demands. For the design of this simulation system both flexibility and a comfortable programming interface were major objectives. On the one hand, the flexibility allows future extensions, like alternative collision detection and response methods. On the other hand, a clear and flexible programming interface was especially necessary as the TüTex software is an integral part of the Virtual Try-On project (Section 6.3) and the interoperability of the various components from all project partners was a prerequisite for a stable system. In the following we give a summary of the different parts of TüTex and describe their specific purposes.

The physical cloth dynamics in TüTex are calculated by a rotated finite element model using linear elasticity. To simulate cloth as realistically as possible it is also necessary to use the correct material parameters. However, the material parameters for woven textiles show different values in the two orthogonal direc-

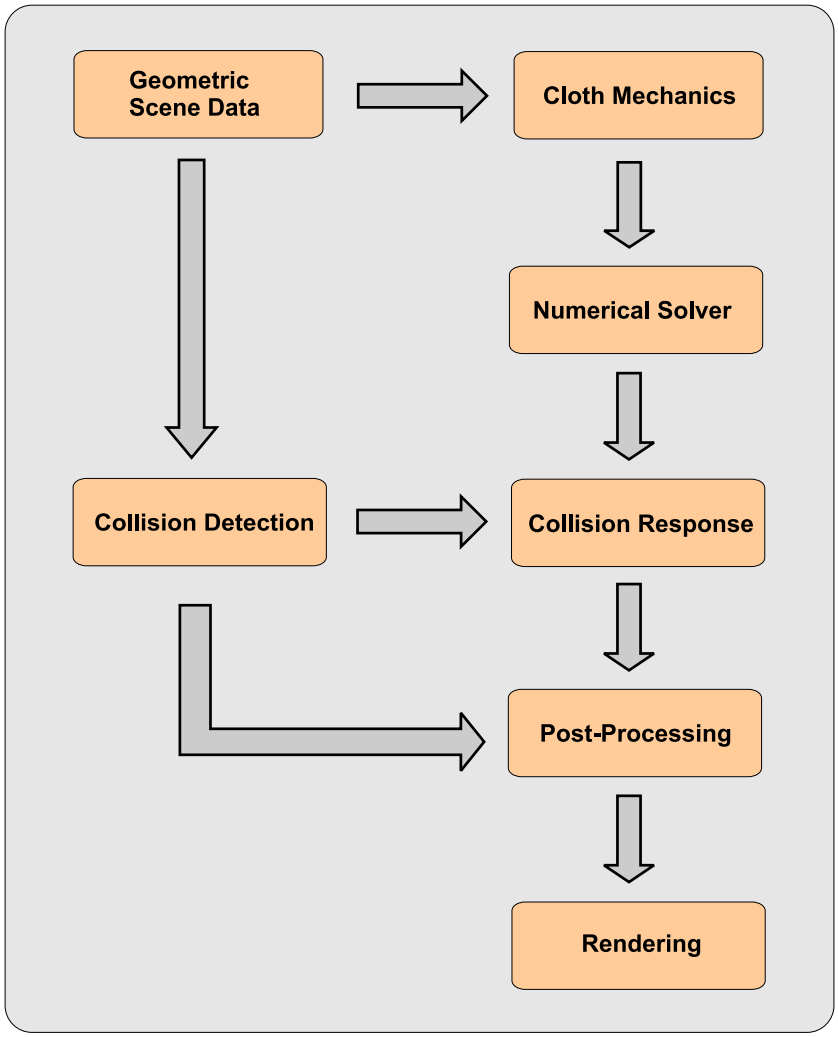


Figure 6.1: Overview of the cloth simulation pipeline in TüTex.

tions, weft and warp, of the threads. Hence, we employ separate parameters that are obtained using a Kawabata measurement system [75]. We used measurements for the two Young moduli that are a measure for the tension resistance. Additionally the shear and bending moduli, the Poisson coefficient, a measure for the transversal contraction, and the mass density control our simulation. All these textile parameters are read into the simulator using XML files. In addition to the internal forces, external forces like gravity and air resistance need to be included into the physical model. While gravity is easy to implement, wind and lee effects as well as the air resistance are incorporated by two aerodynamic models based on particle tracing and the Navier-Stokes equations [3]. Different explicit and implicit numerical solvers were implemented into our system [64, 63, 50]. As high

dimensional linear equation systems have to be solved in this context, the sparsity of the corresponding matrices is exploited in order to yield acceptable computation times. A parallelized time integration scheme for TüTex has been developed by Keckeisen et al. [77].

The collision detection used in TüTex is based on k-DOP hierarchies and is extended by stochastic methods as detailed in Section 2.3. Impulses and constraint-based methods are used to resolve occurred collisions in the system (Chapter 3). All the post-processing methods presented in Chapter 4 and 5 are used to improve the simulated meshes. These methods are only applied for the time steps that are either passed to the rendering or exported as geometry files. The post-processing uses the geometric scene data, i.e. avatars or other rigid collision objects, to allow a collision-free output. Animations of avatars can therefore be imported into TüTex as a keyframed mesh sequence that is linearly interpolated in order to generate in-between frames depending on the simulation time step size. The cloth itself is also part of the geometric scene data and is provided as planar patterns, given as triangulated meshes, together with the corresponding patterns pre-positioned around the avatar. These planar patterns are used as a reference to compute the internal forces based on the rest state of the textiles. Different cloth patterns belonging to the same garment can be sewed together based on sewing information provided in XML files [76]. After a simulation step is finished, the result of the simulation engine can directly be passed to a rendering engine or exported as a sequence of triangulated meshes. Further examples of garment simulations of different clothes are presented in the following sections. As previously shown in this thesis, TüTex is not only restricted to clothing, but also allows the simulations of other cloth objects like table cloths or ribbons (Figure 3.1 and 3.2).

6.3 Virtual Try-On

Over the last years one main focus in the development of TüTex was its integration into the research project Virtual Try-On [4]. The idea behind this project was the realization of two different scenarios. First, for customers both in an Internet shop or in a real boutique it should be possible to try-on garments in a virtual environment. Therefore, the customer is represented by a three-dimensional virtual counterpart, an avatar. Then the customer should be able to choose pieces of clothing, variable both in design and material. If he is convinced by the look and fit he should then be able to buy or order it. This strategy might lead to a more comfortable shopping experience and to smaller return rates for mail order companies. The second scenario is the sale of made-to-measure garments supported by a virtual try-on system. This scenario can be realized by an avatar in com-

bination with adaptable garments represented by cloth patterns. Especially for made-to-measure clothing a reduction of the return rates is of substantial interest, as this clothing cannot be sold to other people.

For both presented scenarios it was necessary to realize different tasks:

- acquire a virtual counterpart of the customer,
- support a interface to CAD software for garment design and pattern construction,
- compute the realistic draping behavior for various materials and give feedback of the fitting of different sizes and cuts,
- visualize the achieved cloth simulation with the correct materials.

In the following we give an overview of this simulation pipeline. A more detailed description of the project and the tasks accomplished by our project partners can be found in [17, 126].

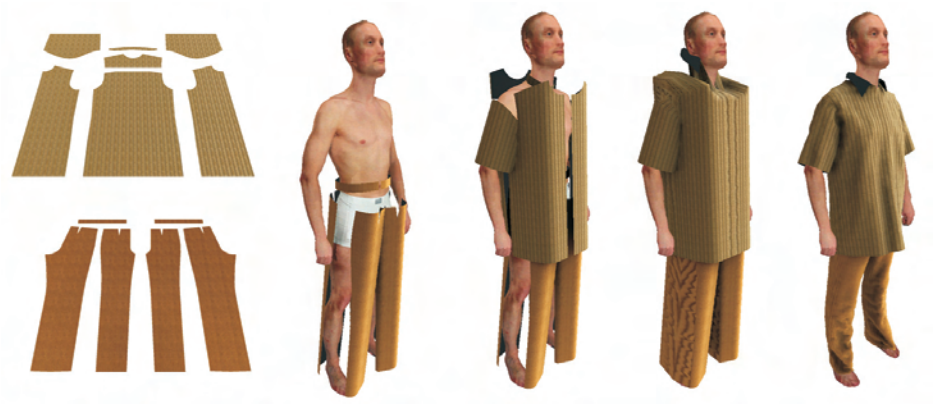


Figure 6.2: The pre-positioning and simulation steps of the pipeline realized in the bmb+f project Virtual Try-On.

The customer's three-dimensional avatar is acquired using a laser scanner. Therefore, the customer has to stand into a kind of dressing room, and within seconds the scan is completed. Textures of the customer's skin are obtained using cameras. For the cloth simulation, first the CAD data of the garments needs to be enriched by information about the seams to assure the correct sewing of the cloth patterns. The exact material parameters for different types of cloth are acquired using a Kawabata measurement system. As starting position for the draping simulation, TüTex utilizes the cloth patterns automatically placed around the avatar

(Figure 6.2). This pre-positioning process is necessary to allow a complete automatization of the simulation pipeline without user interaction. It mimics human knowledge about where to place e.g. trousers or shirts on the body. Based on all the above input data TüTex calculates the correct draping with specific folds and wrinkles. The calculated triangular meshes are then passed to the rendering module, which visualizes the cloth using bidirectional texture functions and self-shadowing.

With this simulation pipeline we were able to realize the ambitious goals of the Virtual Try-On project [12, 13, 14, 16, 17]. We can simulate various types of garment based on CAD data and physical material parameters (Figure 6.3). Using the physically-based simulation, we obtain a realistic prediction of the fit and draping of the garment, as shown in Figure 6.4, where two women with different body size are wearing the same dress. The deformation of the textile and therefore also the forces acting on the body are color-coded from blue to red, representing low to high tension forces. This visualization can be used to evaluate the fit of virtual clothing or to select the best-fitting cloth size out of different alternatives. To realize made-to-measure clothing, it is possible to change the size of the garment patterns, e.g. the length of sleeves or legs, individually. This makes the manufacturing of custom-made clothing possible after the garment has been simulated on the computer. A more complex example is the try-on of different layers of cloth over each other. In Figure 6.5 a blouse is worn over a pair of trousers. This simulation was calculated using a sequential approach for the simulation, i.e. first the underlying layer, here the pair of trousers, and then the layer on top is simulated.

6.4 TüTex Maya Plugin - tcCloth

The TüTex simulation software cannot only be used as stand-alone software or as part of the Virtual Try-On simulation pipeline, it has also been integrated into Alias Maya as a plugin named *tcCloth* [2]. Maya provides a comfortable interface to set up the scenes for the cloth simulation and to use the integrated renderers to rapidly visualize the achieved results. The generation of cloth objects within tcCloth is very intuitive, since the user can easily design planar cloth patterns by drawing closed boundary curves. Thereafter, several patterns can be sewed together by defining seam-lines in between them. A single cloth pattern or multiple connected cloth patterns can then be transformed into a piece of garment. The necessary pre-positioning of the patterns around an avatar is done interactively using the provided moving, bending and rotating tools in Maya (Figure 6.6 (a)).

The implementation of the plugin as well as the software interfaces to TüTex

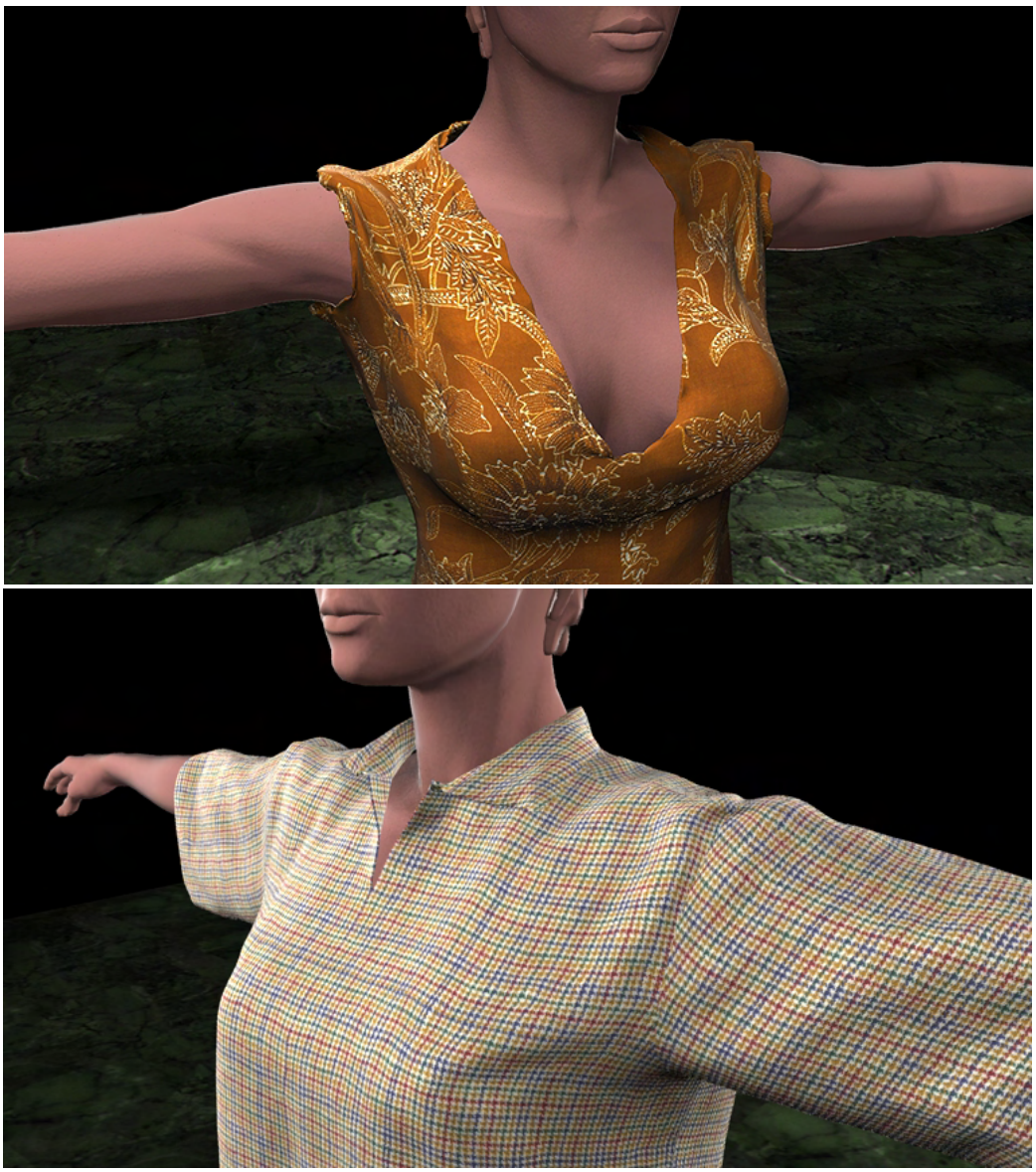


Figure 6.3: Physically-based simulations of a dress and a blouse using TüTex based on CAD data and the corresponding material parameters.

are implemented in C++. User interfaces, like the panels to set material properties or simulation parameters, are realized using the Maya Embedded Language (MEL). This design allows to set all relevant input parameters of TüTex using the associated Maya user interface. For instance, it is possible to set the time step size or to choose a collision detection and response method (Figure 6.6 (b)).

Figure 6.4: A dress tried-on by two women with different body proportions. The internal forces are visualized with colors from red to blue representing high to low tension forces. For the woman in the upper row, low tension forces result, i.e. the dress fits well. It is however to small for the woman in the lower row.

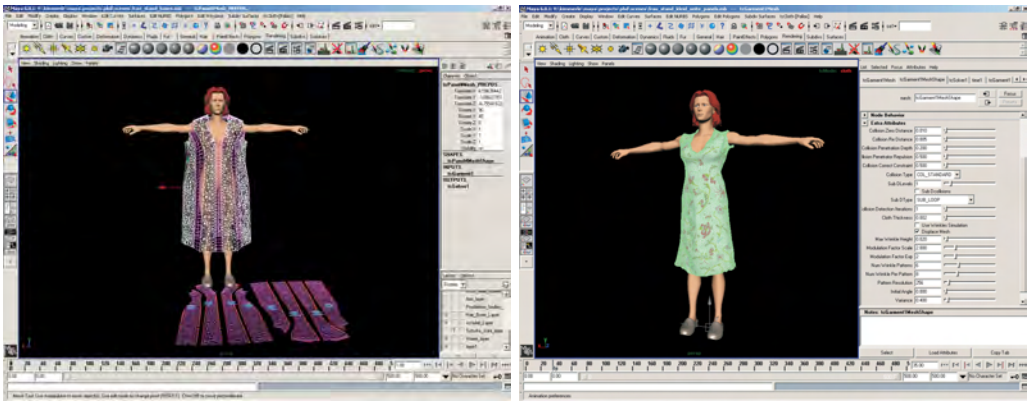
The post-processing methods presented in this thesis are also part of the plugin. Hence, tcCloth is a very powerful tool to test the presented or newly developed algorithms and to generate demo scenes and videos (Figure 6.7). In addition to our post-processing methods, tools included in Maya like mesh extrusion can also be used to enhance the three-dimensional impression of cloth. The simulation results can then be rendered with any renderer running in Maya like Pixar's RenderMan [103] or mental ray of mental images [94].

6.5 Summary

In this chapter, we have given an overview of the TüTex cloth simulation system, its application in the bmb+f research project Virtual Try-On and its integration

Figure 6.5: Different garment layers are simulated sequentially. First, the underlying layer, here the pair of trousers, then the layer on top is simulated.

into the Maya plugin tcCloth. Beside numerical solvers and physical models, collision detection and post-processing methods are an essential part of cloth simulation software. Therefore, all the algorithms presented in this thesis were integrated into the versatile and fast simulation framework system TüTex. Because for custom-made clothing physically-based simulations of cloth based on the real material parameters are required, TüTex became an integral part of the pipeline for Virtual Try-On. With Virtual Try-On, we were the first who realized a cloth simulation system that automatically simulates garment given the CAD data of the cloth patterns, the physical material parameters and a body scan of a customer. The results achieved within this project are very promising, as positive feedback of textile industry and possible customers at the Cebit 2004 [16] has shown, where the complete system was presented. To be further able to efficiently create animations and test scenes, we integrated Tütex with all collision detection and post-processing algorithms into an Alias Maya plugin. This software is currently being prepared to be made freely available to the public.



(a) (b)

Figure 6.6: Interactive design, sewing, and simulation of cloth in the Maya plugin tcCloth. First, the patterns are designed and seam lines are defined (a), then the garment can be simulated and post-processed (b).

Figure 6.7: TüTex cloth simulations computed with the Maya plugin tcCloth.

CHAPTER 7

Conclusions and Future Work

The objective of this thesis was to develop collision detection, collision response and post-processing algorithms for the cloth simulation system TüTex. In the following we first summarize our specific contributions to these fields, before we address possible research directions for the future.

Collision Detection and Response: For collision detection methods employed for deformable objects we presented the state of the art and extensively discussed the respective advantages and disadvantages. We gave a decision matrix that allows to choose the method that fits best to the specific needs of a problem or an application. Using this decision matrix, we motivated the choice of bounding volume hierarchies for the cloth simulation system TüTex. As bounding volume hierarchies per se do not allow a balancing of speed and quality during the collision detection process, we developed an innovative stochastic collision detection method. This hierarchy accelerated stochastic collision detection shows a significantly better performance than a pure stochastic or a pure bounding volume hierarchy approach. To motivate the balancing of speed and quality, we showed that it is not necessary to detect all collisions in order to obtain stable cloth simulations. To resolve emerged collisions we developed an impulse-based response method that is able to handle complex collision and self-collision situations both

for static and for dynamic collision objects. Additionally, we showed how the applied impulses of this method are distributed on the single mesh vertices in order to obtain continuity across mesh elements.

Post-Processing: Since cloth simulation with high resolution meshes is very time consuming, we proposed the simulation of low resolution meshes combined with a subsequent post-procession. To the field of post-processing of virtual textiles we contributed two different approaches. To overcome the problem of the visible polygonal structure of coarse meshes, we proposed to use subdivision methods for triangular meshes and compared the approximating modified Loop and the interpolating modified Butterfly method with respect to their applicability for virtual cloth. As the refinement of the mesh using subdivision methods may cause collisions between the altered cloth mesh and its environment, we combined the subdivision methods with a continuous collision detection and response. The second post-processing approach we introduced in this thesis are the strain-based wrinkle textures. Since coarse meshes cannot model fine folds and wrinkles, these details are added by textures. Based on the assumption of area conservation within a triangle and the strain tensor as a deformation measure, a texture representing a height field over the triangles is generated. Contrary to earlier approaches we are able to automatically generate these wrinkle textures without user interaction, making it applicable for automatic simulation systems like TüTex. Furthermore, we are the first who combined these textures with a collision-free subdivision step in order to realize displacement mapping.

TüTex and Virtual Try-On: All the collision detection and post-processing methods presented in this thesis were included into the cloth simulation engine TüTex. With Virtual Try-On the first system was realized that allows the physically-based simulation of garment based on CAD cloth patterns, physical material parameters and 3D body scans. To be able to easily set up scenes for cloth simulation and to rapidly visualize the achieved results, TüTex was integrated into Alias Maya by converting it to a plugin.

Although the collision detection and post-processing in TüTex was strongly improved using the presented methods, many interesting research directions remain for the future. In the field of collision detection it might be interesting to further develop bounding volume hierarchies so that they efficiently support topology changes. These topology changes arise for example in the design of virtual garments when cloth is cut by scissors. Another important task might be the combination of the presented hierarchy accelerated stochastic collision detection with continuous collision detection in order to avoid tunneling effects. Additionally, the design of collision response methods more suited for detection ratios smaller than 100% seems promising. A very challenging task is the development

of algorithms that allow the detection of collisions on parallel computer systems. Since in these systems, the simulation of virtual cloth is distributed to different computers particularly the detection of self-collisions is complicated because not all necessary data might be present on each of them. An interesting task in the simulation of textiles is the development of a rapid prototyping system for the design of clothing. As the simulation of garment in TüTex is physically-based and thus very realistic, for the textile industry it might be advantageous if expensive prototypes can be replaced by computer simulations. In the context of virtual try-on systems another attractive direction of research, possibly not only in computer graphics, might be the development of systems where the simulated textile is not only perceived visually but also using tactile interfaces.

Authored or Co-Authored Publications

Articles in Conference Proceedings and Journals

[1] O. Etzmuß, M. Keckeisen, S. Kimmerle, J. Mezger, M. Hauth, and M. Wacker. A cloth modelling system for animated characters. In *Proceedings of Graphiktag*, pages 77–86, 2001.

[2] M. Gruber, C. Michel, S. Pabst, M. Wacker, M. Keckeisen, and S. Kimmerle. tccloth - an interactive cloth modeling and animation system. In *Proceedings of Graphiktag*, 2004.

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

[4] S. Kimmerle, M. Keckeisen, J. Mezger, and M. Wacker. TüTex: A cloth modelling system for virtual humans. In *Proceedings 3D Modelling*, 2003.

[5] S. Kimmerle, M. Nesme, and F. Faure. Hierarchy accelerated stochastic collision detection. In *Proceedings of Vision, Modeling, Visualization*, pages 307–314, 2004.

[6] S. Kimmerle, M. Wacker, and C. Holzer. Multilayered wrinkles textures from strain. In *Proceedings of Vision, Modeling, Visualization*, pages 225–232, 2004.

[7] N. Magnenat-Thalmann, F. Cordier, M. Keckeisen, S. Kimmerle, R. Klein, and J. Meseth. Simulation of clothes for real-time applications. In *Proceedings of Eurographics*, 2004. Tutorial.

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

[9] J. Mezger, S. Kimmerle, and O. Etzmuß. Progress in collision detection and response techniques for cloth animation. In *Proceedings of Pacific Graphics*, pages 444–445, 2002. Poster paper.

[10] 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. In *Proceedings of Eurographics*, pages 119–139, 2004. State of the Art Report.

[11] 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. *Computer Graphics Forum*, 2005. Accepted for publication.

[12] M. Wacker, M. Keckeisen, S. Kimmerle, and J. Mezger. Fortschritte in der Textilsimulation für Fashion on Demand. In *Tagungsband 1. Paderborner Workshop Augmented Reality Virtual Reality in der Produktentstehung*, pages 131–141, 2002.

[13] M. Wacker, M. Keckeisen, S. Kimmerle, W. Straßer, V. Luckas, C. Groß, A. Fuhrmann, R. Sarlette, M. Sattler, and R. Klein. Virtual Try-On: Virtuelle Textilien in der Graphischen Datenverarbeitung. *Informatik Spektrum*, 27(6):504–511, 2004.

[14] M. Wacker, M. Keckeisen, S. Kimmerle, W. Straßer, V. Luckas, C. Groß, A. Fuhrmann, M. Sattler, R. Sarlette, and R. Klein. Simulation and visualisation of virtual textiles for virtual try-on. *Research Journal of Textile and Apparel*, 2005. Accepted for publication.

[15] G. Zachmann, M. Teschner, S. Kimmerle, B. Heidelberger, L. Raghupathi, and A. Fuhrmann. Real-time collision detection for dynamic virtual environments. In *Proceedings of IEEE Virtual Reality*, 2005. Tutorial.

Exhibitions

[16] *Virtual Try-On*. CeBIT 2004, Hannover, Germany, March 18-24, Hall 11, Booth D32.

[17] A. Divivier, R. Trieb, A. Ebert, H. Hagen, C. Gross, A. Fuhrmann, V. Luckas, J. Encarnação, E. Kirchdörfer, M. Rupp, S. Vieth, S. Kimmerle, M. Keckeisen, M. Wacker, W. Straßer, M. Sattler, R. Sarlette, and R. Klein. Virtual try-on: Topics in realistic, individualized dressing in virtual reality, 2004. International bmb+f Status Conference "Virtual and Augmented Reality", Leipzig, Germany, February 19-20.

Bibliography

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

[19] Alias web page. http://www.alias.com/.

[20] G. Baciu and W. S.-K. Wong. Hardware-assisted self-collision for deformable surfaces. In *Proceedings of ACM Symposium on Virtual Reality Software and Technology*, pages 129–136. ACM Press, 2002.

[21] G. Baciu and W. S.-K. Wong. Image-based collision detection for deformable cloth models. In *IEEE Transactions on Visualization and Computer Graphics*, volume 10(6), pages 649–663. ACM Press, 2004.

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

[23] S. Bandi and D. Thalmann. An adaptive spatial subdivision of the object space for fast collision detection of animating rigid bodies. In *Proceedings of Eurographics*, pages 259–270, 1995.

[24] D. Baraff. Curved surfaces and coherence for non-penetrating rigid body simulation. *ACM Computer Graphics*, 24(4):19–28, 1990.

[25] D. Baraff and A. Witkin. Large steps in cloth simulation. In *Proceedings of ACM SIGGRAPH*, pages 43–54, 1998.

[26] D. Baraff, A. Witkin, and M. Kass. Untangling cloth. In *Proceedings of ACM SIGGRAPH*, pages 862–870, 2003.

[27] R. Barzel, J. Hughes, and D. N. Wood. Plausible motion simulation for computer graphics animation. In *Proceedings of Computer Animation and Simulation*, pages 183–197, 1996.

[28] J. F. Blinn. Simulation of wrinkled surfaces. In *Proceedings of ACM SIGGRAPH*, volume 12(3), pages 286–292, 1978.

[29] J. Bonet and R. D. Wood. *Nonlinear continuum mechanics for finite element analysis*. Cambridge University Press, 2000.

[30] J. W. Boyse. Interference detection among solids and surfaces. *Communications of the ACM*, 22(1):3–9, 1979.

[31] D. E. Breen, D. H. House, and M. J. Wozny. A particle-based model for simulating the draping behavior of woven cloth. *Textile Research Journal*, 64(11):663–685, 1994.

[32] D. E. Breen, S. Mauch, R. T. Whitaker, and J. Mao. 3d metamorphosis between different types of geometric models. In *Proceedings of Eurographics*, pages 36–48, 2001.

[33] R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. In *Proceedings of ACM SIGGRAPH*, pages 594–603, 2002.

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

[35] E. Catmull and J. Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. In *Computer Aided Design*, volume 10, pages 350–355, 1978.

[36] K. Choi and H. Ko. Stable but responsive cloth. In *Proceedings of ACM SIGGRAPH*, pages 604–611, 2002.

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

[38] J. D. Cohen, M. C. Lin, D. Manocha, and M. Ponamgi. I-COLLIDE: An interactive and exact collision detection system for large-scale environments. In *Symposium on Interactive 3D Graphics*, pages 189–196, 1995.

[39] J. Combaz and F. Neyret. Painting folds using expansion textures. In *Proceedings of Pacific Graphics*, pages 176–183, 2002.

[40] R. L. Cook. Shade trees. In *Proceedings of ACM SIGGRAPH*, volume 18(3), pages 223–231, 1984.

[41] 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.

[42] T. DeRose, M. Kass, and T. Truong. Subdivision surfaces in character animation. In *Proceedings of ACM SIGGRAPH*, pages 85–94, 1998.

[43] M. Desbrun, P. Schröder, and A. Barr. Interactive animation of structured deformable objects. In *Proccedings of Graphics Interface*, pages 1–8, 1999.

[44] M. Doggett, A. Kugler, and W. Straßer. Displacement mapping using scan conversion hardware architectures. *Computer Graphics Forum*, 20:13–26, 2001.

[45] D. Doo and M. Sabin. Analysis of the behaviour of recursive division surfaces near extraordinary points. In *Computer Aided Design*, volume 10, pages 356–360, 1978.

[46] N. Dyn, J. Gregory, and D. Levin. A butterfly subdivision scheme for surface interpolation with tension control. In *ACM Transactions on Graphics*, volume 9, pages 160–169, 1990.

[47] B. Eberhardt, O. Etzmuss, and M. Hauth. Implicit-explicit schemes for fast animation with particle systems. In *Proceedings of Computer Animation and Simulation*, 2000.

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

[49] S. A. Ehmann and M. C. Lin. Accurate and fast proximity queries between polyhedra using convex surface decomposition. *Computer Graphics Forum*, 20(3):500–510, 2001.

[50] O. Etzmuß. *Animation of Surfaces with Applications to Cloth Modelling*. PhD thesis, Universität Tübingen, 2002.

[51] O. Etzmuß, M. Keckeisen, and W. Straßer. A fast finite element solution for cloth modelling. In *Proceedings of Pacific Graphics*, 2003.

[52] S. Fisher and M. Lin. Deformed distance fields for simulation of non-penetrating flexible bodies. In *Proceedings of Computer Animation and Simulation*, 2001.

[53] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones. Adaptively sampled distance fields: A general representation of shape for computer graphics. In *Proceedings of ACM SIGGRAPH*, pages 249–254, 2000.

[54] A. Fuhrmann, G. Sobotka, and C. Gross. Distance fields for rapid collision detection in physically based modeling. In *Proceedings of GraphiCon 2003*, pages 58–65, 2003.

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

[56] S. Gibson. Using distancemaps for smooth surface representation in sampled volumes. In *Proceedings of IEEE Volume Visualization Symposium*, pages 23–30, 1998.

[57] J. Goldsmith and J. Salmon. Automatic creation of object hierarchies for ray tracing. *IEEE Computer Graphics and Applications*, 7(5):14–20, 1987.

[58] S. Gottschalk, M. Lin, and D. Manocha. OBB-Tree: A hierarchical structure for rapid interference detection. In *Proceedings of ACM SIGGRAPH*, pages 171–180, 1996.

[59] N. Govindaraju, S. Redon, M. Lin, and D. Manocha. Cullide: Interactive collision detection between complex models in large environments using graphics hardware. In *Proceedings of ACM Graphics Hardware*, 2003.

[60] L. J. Guibas, D. Hsu, and L. Zhang. H-walk: Hierarchical distance computation for moving convex bodies. In W. V. Oz and M. Yannakakis, editors, *Proceedings of ACM Symposium on Computational Geometry*, pages 265–273, 1999.

[61] S. Guy and G. Debunne. Monte-carlo collision detection. Technical Report RR-5136, INRIA, 2004.

[62] S. Hadap, E. Bangerter, P. Volino, and N. Magnenat-Thalmann. Animating wrinkles on clothes. In *Proceedings of IEEE Visualization*, 1999.

[63] M. Hauth. *Visual Simulation of Deformable Models*. PhD thesis, Universität Tübingen, 2004.

[64] M. Hauth, O. Etzmuß, and W. Straßer. Analysis of numerical methods for the simulation of deformable models. *The Visual Computer*, 19(7–8):581–600, 2003.

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

[66] B. Heidelberger, M. Teschner, and M. Gross. Detection of collisions and self-collisions using image-space techniques. In *Journal of WSCG*, pages 145–152, 2004.

[67] B. Heidelberger, M. Teschner, R. Keiser, M. Müller, and M. Gross. Consistent penetration depth estimation for defomrable collision response. In *Proceedings of Vision, Modeling, Visualization*, pages 339–346, 2004.

[68] J. Hirche, A. Ehlert, S. Guthe, and M. Doggett. Hardware accelerated per-pixel displacement mapping. In *Proccedings of Graphics Interface*, 2004.

[69] K. E. Hoff III, J. Keyser, M. Lin, D. Manocha, and T. Culver. Fast computation of generalized Voronoi diagrams using graphics hardware. In *Proceedings of ACM SIGGRAPH*, pages 277–286, 1999.

[70] C. Holzer. Post-processing bei der simulation virtueller textilien. Diplomarbeit, WSI/GRIS, Universität Tübingen, 2003.

[71] D. H. House and D. E. Breen, editors. *Cloth Modeling and Animation*. AK Peters, 2000.

[72] P. M. Hubbard. Collision detection for interactive graphics applications. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):218–230, 1995.

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

[74] M. W. Jones and R. Satherley. Using distance fields for object representation and rendering. In *Proceedings of Eurographics*, pages 37–44, 2001.

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

[76] M. Keckeisen. *Physical Cloth Simulation and Applications for the Visualization, Virtual Try-On, and Interactive Design of Garments*. PhD thesis, Universität Tübingen, 2005.

[77] M. Keckeisen and W. Blochinger. Parallel Implicit Integration for Cloth Animations on Distributed Memory Architectures. In *Proceedings of Eurographics Symposium on Parallel Graphics and Visualization*, 2004.

[78] M. Keckeisen, M. Feurer, and M. Wacker. Tailor tools for interactive design of clothing in virtual environments. In *Proceedings of ACM Symposium on Virtual Reality Software and Technology*, 2004.

[79] Y. Kim, M. Otaduy, M. Lin, and D. Manocha. Fast penetration depth computation for physically-based animation. In *Proceedings of SIGGRAPH Symposium on Computer Animation*, pages 23–31, 2002.

[80] J. Klein and G. Zachmann. Adb-trees: Controlling the error of time-critical collision detection. In *Proceedings of Vision, Modeling, Visualization*, 2003.

[81] J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowrizal, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998.

[82] D. Knott and D. Pai. Cinder: Collision and interference detection in real–time using graphics hardware. In *Proccedings of Graphics Interface*, 2003.

[83] 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), 1998.

[84] S. Krishnan, A. Pattekar, M. C. Lin, and D. Manocha. Spherical shells: A higher-order bounding volume for fast proximity queries. Technical report, Department of Computer Science, University of North Carolina, 1997.

[85] B. Lafleur, N. Magnenat-Thalmann, and D. Thalmann. Cloth animation with self-collision detection. In *Proceedings of the Conference on Modelling in Computer Graphics*, pages 179–187, 1991.

[86] C. Larboulette and M.-P. Cani. Real-time dynamic wrinkles. In *Proccedings of Computer Graphics International*, pages 522–525, 2004.

[87] T. Larsson and T. Akenine-Möller. Collision detection for continuously deforming bodies. In *Proceedings of Eurographics*, pages 325–333, 2001. short presentation.

[88] C. Levinthal. Molecular model-building by computer. *Scientific American*, 214:42–52, 1966.

[89] J. C. Lombardo, M.-P. Cani, and F. Neyret. Real-time collision detection for virtual surgery. In *Proceedings of Computer Animation*, pages 82–91. IEEE CS Press, 1999.

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

[91] C. Loop. Bounded curvature triangle mesh subdivision with the convex hull property. *The Visual Computer*, 18:316–325, 2002.

[92] S. Mauch. *Efficient Algorithms for Solving Static Hamilton-Jacobi Equations*. Phd thesis, Californa Institute of Technology, 2003.

[93] S. Melax. Dynamic plane shifting bsp traversal. In *Proccedings of Graphics Interface*, pages 213–220, 2000.

[94] Mental Images web page. http://www.mentalimages.com/.

[95] B. Mirtich. Efficient algorithms for two-phase collision detection. Technical Report TR-97-23, Mitsubishi Electric Research Laboratory, 1997.

[96] B. Mirtich and J. Canny. Impulse-based simulation of rigid bodies. In *Proceedings of symposium on interactive 3D graphics*, pages 181–188, 1995.

[97] M. Moore and J. Wilhelms. Collision detection and response for computer animation. In *Proceedings of ACM SIGGRAPH*, pages 289–298, 1988.

[98] K. Moule and M. McCool. Efficient bounded adaptive tesselation of displacement maps. In *Proccedings of Graphics Interface*, 2002.

[99] 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.

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

[101] B. A. Payne and A. W. Toga. Distance field manipulation of surface models. *IEEE Computer Graphics and Applications*, 12(1):65–71, January 1992.

[102] K. Perlin. An image synthesizer. In *Proceedings of ACM SIGGRAPH*, volume 19(3), pages 287–296, 1985.

[103] Pixar web page. http://www.pixar.com/.

[104] J. C. Platt and A. H. Barr. Constraint methods for flexible models. In *Proceedings of ACM SIGGRAPH*, pages 279–288, 1988.

[105] H. Prautzsch and G. Umlauf. Improved triangular subdivision schemes. In F. Wolter and N. Patrikalakis, editors, *Proccedings of Computer Graphics International*, pages 626–632, 1998.

[106] X. Provot. Collision and self-collision handling in cloth model dedicated to design garments. In *Proccedings of Graphics Interface*, pages 177–189. Canadian Information Processing Society, 1997.

[107] S. Quinlan. Efficient distance computation between non-convex objects. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3324–3329, 1994.

[108] M. O. Rabin. *Probabilistic algorithms*. Academic Press, 1976.

[109] L. Raghupathi, L. Grisoni, F. Faure, D. Marchal, M.-P. Cani, and C. Chaillou. An intestine surgery simulator: Real-time collision processing and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 2004.

[110] S. Redon, A. Kheddary, and S. Coquillart. Fast continuous collision detection between rigid bodies. In *Proceedings of Eurographics*, pages 279–288, 2002.

[111] N. Roussopoulos and D. Leifker. Direct spatial search on pictorial databases using packed R-trees. In *Proceedings of ACM-SIGMOD International Conference on Management of Data*, pages 17–31, 1985.

[112] P. Schröder and D. Zorin. Subdivision for modeling and animation. In *Proceedings of ACM SIGGRAPH*, 1998. Course.

[113] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Science*, 93(4):1591–1595, 1996.

[114] J. Shade, S. Gortler, L. W. He, and R. Szeliski. Layered depth images. In *Proceedings of ACM SIGGRAPH*, pages 231–242, 1998.

[115] M. Shinya and M. Forgue. Interference detection through rasterization. *The Journal of Visualization and Computer Animation*, 2:132–134, 1991.

[116] C. Sigg, R. Peikert, and M. Gross. Signed distance transform using graphics hardware. In *Proceedings of IEEE Visualization*. IEEE Computer Society Press, 2003.

[117] Softimage web page. http://www.softimage.com/.

[118] A. Sud, M. A. Otaduy, and D. Manocha. Difi: Fast 3d distance field computation using graphics hardware. In *Proceedings of Eurographics*, pages 557–566, 2004.

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

[120] 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*, pages 47–54, 2003.

[121] J. Thingvold and E. Cohen. Physical modeling with b-spline surfaces for interactive design and animation. In *Proceedings of Symposium on Interactive 3-D graphics*, pages 129–137, 1990.

[122] G. Turk. Interactive collision detection for molecular graphics. Technical Report TR90-014, University of North Carolina at Chapel Hill, 1990.

[123] S. Uno and M. Slater. The sensitivity of presence to collision response. In *Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS)*, page 95, 1997.

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

[125] T. Vassilev, B. Spanlang, and Y. Chrysanthou. Fast cloth animation on walking avatars. In *Proceedings of Eurographics*, 2001.

[126] Virtual Try-On project web page. http://www.virtualtryon.de/.

[127] P. Volino, M. Courshesnes, and N. Magnenat-Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. In *Proceedings of ACM SIGGRAPH*, pages 137–144, 1995.

[128] P. Volino and N. Magnenat-Thalmann. Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. *Computer Graphics Forum*, 13(3):155–166, 1994.

[129] P. Volino and N. Magnenat-Thalmann. Collision and self-collision detection: Efficient and robust solutions for higly deformable surfaces. In *Proceedings of Computer Animation and Simulation*, pages 55–65, 1995.

[130] P. Volino and N. Magnenat-Thalmann. Fast geometrical wrinkles on animated surfaces. In *Journal of WSCG*, 1999.

[131] P. Volino and N. Magnenat-Thalmann. Accurate collision response on polygonal meshes. In *Proceedings of Computer Animation*, pages 179–188, 2000.

[132] P. Volino and N. Magnenat-Thalmann. Implementing fast cloth simulation with collision response. In *Proccedings of Computer Graphics International*, pages 257–268, 2000.

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

[134] R. Westermann, L. Kobbelt, and T. Ertl. Real-time exploration of regular volume data by adaptive reconstruction of isosurfaces. *The Visual Computer*, 15(2):100–111, 1999.

[135] J. Wu and L. Kobbelt. Piecewise linear approximation of signed distance fields. In *Proceedings of Vision, Modeling, Visualization*, pages 513–520, 2003.

[136] Y. Wu, P. Kalra, and N. Magnenat-Thalmann. Physically-based wrinkle simulation and skin rendering. In *Proceedings of Computer Animation and Simulation*, 1997.

[137] G. Zachmann. Rapid collision detection by dynamically aligned DOP-trees. In *Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS)*, pages 90–97, 1998.

[138] G. Zachmann. Minimal hierarchical collision detection. In *Proceedings of ACM Symposium on Virtual Reality Software and Technology*, pages 121–128, 2002.

[139] D. Zhang and M. Yuen. Collision detection for clothed human animation. In *Proceedings of Pacific Graphics*, pages 328–337, 2000.

[140] D. Zorin, P. Schröder, and W. Sweldens. Interpolating subdivision for meshes with arbitrary topology. In *Proceedings of ACM SIGGRAPH*, pages 189–192, 1996.

Lebens- und Bildungsgang

Stefan Kimmerle

3. November 1975	geboren in Nürtingen
1982 - 1985	Grundschule Neuffen
1985 - 1994	Hölderlin Gymnasium Nürtingen, Abschluss Abitur
1994 - 1995	Studium der Physik an der Universität Stuttgart
1995 - 2000	Studium der Physik mit den Schwerpunkten Halbleiterphysik und Medizinische Physik an der Eberhard-Karls-Universität Tübingen
07/1997 - 06/1998	Auslandsstudium an der San Diego State University im Rahmen eines Stipendiums der Universität Tübingen
10/1999-10/2000	Diplomarbeit *Charakterisierung und Modellierung mikromechanischer Beschleunigungssensoren zur Parameterextraktion* bei der Robert Bosch GmbH, Reutlingen, unter Betreuung von Prof. Dr. Dieter Kern
10/2000	Abschluss des Studiums als Diplom-Physiker
seit 01/2001	Wissenschaftlicher Mitarbeiter am Lehrstuhl für Graphisch-Interaktive System von Prof. Dr.-Ing. Dr.-Ing. E.h. Wolfgang Straßer des Wilhelm-Schickard Instituts für Informatik der Universität Tübingen
07/2003 - 10/2003	Gastwissenschaftler am INRIA Rhône-Alpes, Grenoble, bei Prof. Dr. Marie-Paule Cani und Prof. Dr. François Faure
07/2004 - 10/2004	Gastwissenschaftler am INRIA Rhône-Alpes als Stipendiat des Deutschen Akademischen Austausch Dienstes (DAAD)