# Improved Collision Detection and Response Techniques for Cloth Animation

Johannes Mezger, Stefan Kimmerle, Olaf Etzmuß

WSI-2002-5
August 2002

Graphisch-Interaktive Systeme
Wilhelm-Schickard-Institut
Universität Tübingen
D-72076 Tübingen, Germany
e-mail: {mezger,kimmerle,etzmuss}@gris.uni-tuebingen.de
WWW: http://www.gris.uni-tuebingen.de


ISSN 0946-3852

# Improved Collision Detection and Response Techniques for Cloth Animation

Johannes Mezger, Stefan Kimmerle, Olaf Etzmuß
WSI/GRIS
University of Tübingen
Sand 14, 72076 Tübingen, Germany
email: {jmezger,kimmerle,etzmuss}@gris.uni-tuebingen.de

## Abstract

In the animation of deformable objects, collision detection and response are crucial for the performance. Contrary to volumetric bodies, the accuracy requirements for the collision treatment of textiles are particularly strict because any overlapping is visible. Therefore, we apply methods specifically designed for deformable surfaces that speed up the collision detection.

In this paper the efficiency of bounding volume hierarchies is improved by adapted techniques for building and traversing these hierarchies. An extended set of heuristics is described that allows to prune the hierarchy. Oriented inflation of bounding volumes enables us to detect proximities with a minimum of extra cost. Eventually, the distance of the mesh faces is computed accurately, and constraints respond to the collisions.

**CR Categories:**

I.3.5 [Computational Geometry and Object Modeling]: Object hierarchies, Physically based modeling

I.3.7 [Three-Dimensional Graphics and Realism]: Animation

**Keywords:**

Cloth Simulation, Collision Detection, Collision Response.

## 1 Introduction

A physically correct cloth simulation requires collision avoidance and therefore an effectively robust detection system. Each penetration violates reality and often results in expensive correction procedures. As collision detection has to be performed at discrete points of the simulation time, the size of the simulation time step must be limited such that collisions can be correctly detected and resolved in between.

Since much progress has been achieved in improving the numerical solution, most animations employ large time steps for fast simulations. However, large time steps make the collision detection and response more difficult because the movement during one time step can be significant. The best solution to accommodate this is the early detection of collisions in a specified collision region around the object. Collision detection algorithms must be extended to detect such proximity also. Moreover, the collision response schemes need to adapt to large time steps and avoid interpenetration of the collision objects while still allowing realistically close distances between them.

In this paper we employ the notion of object-based hierarchies, first applied to cloth modelling by Volino et al. [28]. The hierarchical representations of all objects, including the deformable surfaces of arbitrary meshed textiles are built in a preprocessing step. We will study and evaluate different techniques to improve the hierarchy generation and to speed up the updating and traversal of the trees. In order to save computation time, several heuristics are used to prune the trees, including curvature and coherence criteria.

As not only collisions but also proximities are to be detected, the bounding volumes are inflated. In order to minimize additional overlapping of the

bounding volumes, the inflation is oriented in the direction of high velocity.

Accurate collision response is only possible with a measure of the distance between the mesh faces. There are several algorithms that can compute this distance with only very few iterations. Using the distance of the faces, an adaptive, constraint-based collision response approach is employed to avoid interpenetrations. To model resting contacts between close objects, the collision response is scaled by the relative velocity of the colliding faces.

# 2 Previous Work

Many collision detection methods for various purposes have been developed in the past [22]. Some of them are employed and adapted for the particular requirements of cloth modelling.

Collision detection for convex polyhedra has been extensively studied and is based on the GJK-Algorithm [12], Lin-Canny-Algorithm [21] or V-Clip [24]. Non-convex objects can be decomposed into convex parts [10, 9]. $R$-trees [14] provide the theoretical basics for bounding volume hierarchies [3, 13, 20, 19, 16, 26], which are mostly used to generate hierarchical representations of complex meshes. In addition, possibly colliding objects are identified by Sweep-and-Prune strategies [5]. As opposed to bounding volume hierarchies, regular grids partition the scene into voxels [4, 32]. Alternatively, graphics hardware [1] can be employed to detect collisions in image-space, which was even investigated for cloth modelling [27].

More independency of the detection costs of the complexity of the objects is achieved with implicit representations, as the potential of implicit surfaces provides a straightforward collision test for the approximated objects [7, 8].

Particular advances in accelerating the self-collision detection are achieved by Volino et al. [28]. They use a region-merge algorithm to build hierarchies on top of a polygonal mesh, storing adjacency information for the regions. The region normals are sampled to determine the curvature of a region and to reject self-intersections. They also introduce a technique that observes the history of close regions to guarantee a consistent collision response [6]. Recent publications [29] additionally address $k$-DOPs as bounding volumes. Provot [25] describes a similar approach for the surface curvature heuristic, which we extend in our system. Johnson et al. [18] show how normal cone hierarchies can accelerate not only distance computations, but also lighting and shadowing.

Collisions between deformable objects are much more difficult to treat than collisions between rigid objects, because a response for each face or particle has to be computed and care must be taken not to introduce additional stiffness. Therefore, several collision response schemes for cloth animation have been presented. While Baraff and Witkin [2] use a constraint based approach, Volino et al. [30] set position, velocity, and acceleration for the colliding particles. Volino [30] and Provot [25] also propose techniques for handling multiple collisions.

# 3 Bounding Volumes

In complex dynamic scenes, bounding volumes have to be permanently readapted to the approximated geometry. Arbitrary object deformations cannot be expressed by rigid rotations and translations, thus orientable bounding volumes like OBBs [13], QuOSPOs [16], or Dynamically Aligned DOPs [31] are not suitable for cloth modelling. We choose common $k$-DOPs [19] as they show better convergence than spheres [17] or ordinary AABBs [26].

The advantages of a bounding volume hierarchy of $k$-DOPs over other hierarchies are exposed in section 4.

## 3.1 $k$-DOPs

A $k$-DOP[19] (discrete oriented polytope) is a convex polyhedron defined by $k$ halfspaces denoted as

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

The normals $n_i$ of the corresponding hyperplanes of all $k$-DOPs are discrete and form the small set $N = \{n_1, \ldots, n_k\} \subseteq \mathbb{R}^m$. For arithmetic reasons the entries of the normal-vectors are usually chosen from the set $\{-1, 0, 1\}$. In order to turn the intersection test for the polyhedrons into simple interval tests, the hyperplanes have to form $k/2$ parallel pairs. E.g. an axis aligned bounding box (AABB, 6-DOP) in $\mathbb{R}^3$ is given by $N_6 =$

$\{(\pm1,0,0),(0,\pm1,0),(0,0,\pm1)\}$, an octahedron (8-DOP) is generated by setting all normal components to $\pm1$. We usually use 14-DOPs ($N_{14} = N_6 \dot\cup N_8$) or 18-DOPs (AABB with clipped edges).

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

### 3.2 $k$-DOP Inflation

In order to use rather large time steps for the simulation, not only real collisions but also object proximities have to be detected. Let $\varepsilon_{close}$ be the maximum distance of two meshes where proximities have to be detected, depending on the velocities of the vertices and the time step size. Enlarging the $k$-DOPs by an offset $\varepsilon_{close}/2$ in each of its $k$ directions turns the usual overlap test into proximity detection. It easily can be verified that the overlap of such two enlarged $k$-DOPs is a necessary condition for actual $\varepsilon_{close}$–proximity.

### 3.3 Oriented $k$-DOP Inflation

The unoriented inflation implies a higher degree of self-overlapping between contiguous bounding volumes. Thus, the number of overlap tests severely increases depending on the amount of inflation. For this reason, the unoriented inflation is restricted to close proximities and cannot be used to detect potential collisions among objects with higher relative velocities.

To retrieve collisions within the movement of the objects between two frames, the bounding volumes have to enclose the space which is likely to be traversed. To determine this space, the next time step size and the velocity of the vertices have to be estimated. Then, the new vertex positions can be extrapolated and the bounding volumes can be readapted to inclose the old as well as the new vertices. But, as this method would at least double the cost of updating the leaves of the bounding volume hierarchy, we introduce the *oriented k-DOP inflation* as shown in figure (1).

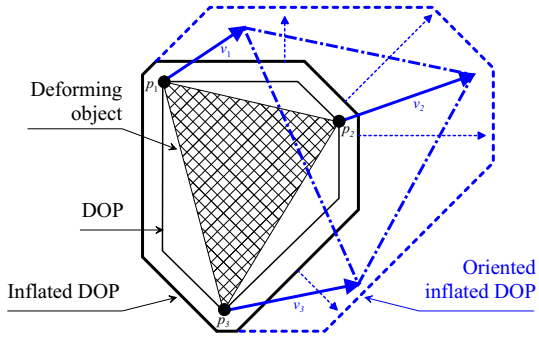



Figure 1: Estimated movement and oriented inflation of the 8-DOP.

The oriented inflation updates each of the $k/2$ intervals depending on the normalized mean axis $\overline{v}$ and the maximal velocity $\hat{v}$ of the velocity cone (section 5.2). The interval limits are increased by the distance

$$d_i = \varepsilon_{close}/2 + \max(\langle \overline{v}, n_i \rangle \cdot \hat{v} \cdot \triangle t,\ 0), \qquad (1)$$

$n_i$ denoting the normal of the hyperplane and $\triangle t$ the expected time step size. At least $k/2$ of the normal vectors do not point into the movement direction, resulting in $\langle \overline{v}, n_i \rangle \leq 0$. If the velocity cone has no principal direction of movement ($\alpha \gg 0$), the ordinary inflation by

$$d = \max(\varepsilon_{close}/2,\ \hat{v} \cdot \triangle t)$$

is applied.

## 4 Dynamic $k$-DOP-Hierarchy

Although voxel-based methods like regular grids can be useful for collision detection and even cloth modeling [32], they do not support the detection of proximities and therefore are not acceptable for the large time step sizes of implicit solvers. Moreover, object based heuristics which prune the collision test for whole parts of the scene cannot operate on voxels.

The dynamic approximation of meshes by implicit surfaces provides very fast particle–surface tests, but the simulation then depends on the resolution of the textiles, and an efficient self-collision detection can barely be realized.

Graphics-hardware based methods [27] are hardware-dependent and cannot solve the self-collision detection problem either. As they generally return rather inexact distances, an accurate collision response remains difficult.

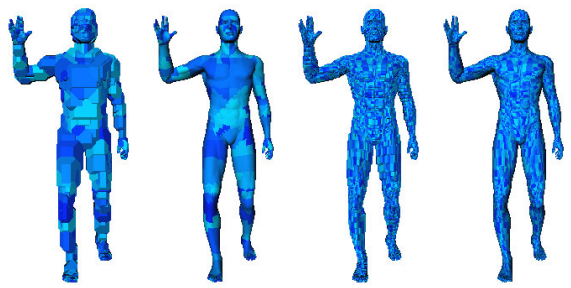
Therefore, a realistic cloth modelling system requires bounding volume hierarchies to be robust and efficient at the same time. We propose to combine the advantages of a top-down $k$-DOP hierarchy with a surface curvature criterion.

## 4.1 Hierarchy Generation

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

$$BV_k(V) = \widetilde{\bigcup}_{p \in P(V)} BV_k(p) \tag{2}$$

for a set of vertices $V$ and an arbitrary partition $P(V)$. Hence, the optimal bounding volume for a node in the hierarchy can be easily computed by merging its child bounding volumes. Vice versa the hierarchy can be efficiently built using a top-down splitting method. Figure (2) shows two hierarchy levels for the 18-DOP-hierarchy of an avatar.



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

Figure 2: Two levels of an 18-DOP-hierarchy. (a) and (c) show the 18-DOPs, (b) and (d) the corresponding regions on the surface.

In contrast to bottom-up methods [28], the initial geometry fits well in the bounding volumes because the faces of a region are selected such that they correspond with the shape of the bounding volume. However, dynamic meshes may of course lose this property when movements other than translations occur.

## 4.2 Node split

The bounding volumes are split according to the longest side. In our implementation the longest side of a $k$-DOP is determined by the face pair with the maximum distance. The $k$-DOP is split parallel to this face pair through its center. As generally some polygons are cut by the splitting plane, they are assigned to that child node which would contain the smallest number of polygons. In the lower hierarchy levels, if all polygons happen to be cut, each of them is assigned to its own node. Finally, as the corresponding vertices for the node are known, the $k$-DOPs can be optimally fitted to the underlying faces. Although this method is simple, it turns out to be efficient on the one hand and to produce well balanced trees on the other hand. The complete hierarchy setup for objects holding several thousands of polygons can be performed within merely a second, allowing to add objects dynamically to the scene. To achieve optimal collision detection performance, the splitting continues until one single polygon remains per leaf.

## 4.3 Lazy Hierarchy Update

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 (equation 2). Parts of the hierarchy where vertices do not traverse more than a distance $b$, $b < \varepsilon_{close}/2$, can be omitted during the hierarchy update for a time $\hat{t} = b/\hat{v}$, if proximities smaller than $\varepsilon_{close} - 2b$ are to be detected, $\hat{v}$ denoting the maximum speed of the vertices (figure 3).

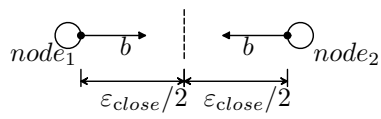



Figure 3: Tolerance distance for the lazy hierarchy update.

Thus, the hierarchy update is accelerated for slow parts of the scene and for small time step sizes.

### 4.4 Trees

Previous approaches employed binary trees to store the hierarchy since they require the smallest number of overlap tests. However, the depth and number of nodes are maximal, and consequently the recursion during overlap tests is deeper than for any higher order tree.

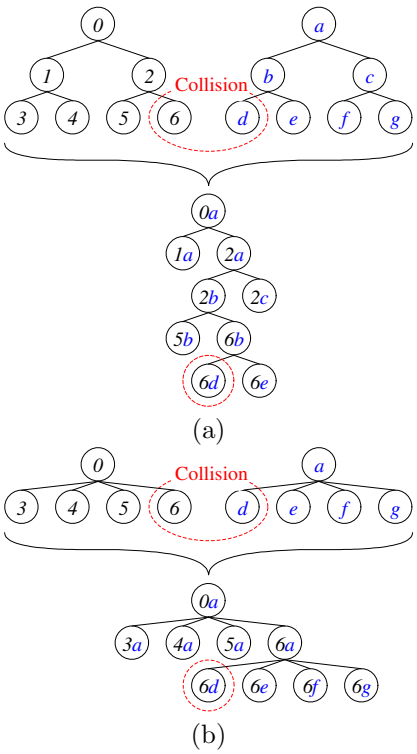



Figure 4: Recursion using binary trees (a) and quadtrees respectively (b).

Figure (4) shows the reduction of recursion depth for detecting two overlapping leaves by equivalent quadtrees instead of binary trees. Note that in this case the recursion depth is reduced by the factor 2, whereas the number of overlap tests remains equal. However, if only the root nodes overlap in the example, the quadtrees require four overlap tests, which is two times more than using binary trees. Since overlap tests for $k$-DOPs only need $k/2$ interval tests in the worst (overlapping) case, a slight increase of overlap tests is acceptable. Our implementation is able to use arbitrary $2^n$-trees, but quadtrees and octrees have turned out to be the fastest.

## 5 Heuristics

In collision detection, heuristics can speed-up the hierarchy update and the intrinsic collision test. However, resulting errors have to be limited strictly in order to preserve the accuracy of the entire collision detection.

We use two different data structures ("cones") that represent both a principal direction and a measure for the correlation of a set of vectors.

### 5.1 Normal Cones

A very exact method to reject possible self-intersections for a certain region was suggested by Volino and Magnenat-Thalmann [28], where a vector is searched that has positive dot product with all normals of the region. If such a vector exists and the projection of the region onto a plane in direction of the vector does not self-intersect, the region cannot self-intersect either.

In our system we employ Provot's method [25], which is very fast and accurate enough for regions having a sufficiently convex border. The $k$-DOP regions generated by our hierarchy setup usually meet this condition, and moreover we are able to extend easily the idea to the detection of self-proximities. For every region a cone is maintained representing a superset of the normal directions. The cone can be calculated during the bottom-up hierarchy update by very few arithmetic operations. The apex angle $\alpha$ of the cone represents the curvature of the region, indicating possible intersections if $\alpha \geq \pi$. In order to detect proximities as well, we replace this intersection criterion by the self-proximity criterion $\alpha \geq \varepsilon_{closeAngle}$ for an angle $\varepsilon_{closeAngle} \leq \pi$. It turned out that the choice of $\varepsilon_{closeAngle}$ is not crucial. It just has to be decreased if the simulation allows rather spiky bends.

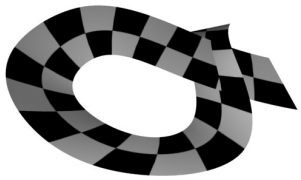


Figure 5: Self-intersecting mesh with correlated normals but concave shape.

Still there remains the problem that hierarchy regions can have severe non-convex shape and therefore compromise the robustness of the surface curvature criterion. Figure (5) shows such a surface that self-intersects although the apex angle of its normal cone is rather small. We divide such a mesh into several face groups and build an adjacency matrix for the groups. The curvature heuristic is not applied to non-adjacent groups during the self-collision test. Thus, collisions of faces are surely detected if they are separated on the surface by at least one group.

The groups also play an important role in the optimization of the primitive pair test (section 6.2).

### 5.2 Velocity Cones

We propose a new heuristic designed to prune off those parts of the scene where only small velocities occur. For that purpose we introduce the *velocity cone* (figure 6), which is also used to detect temporal coherence during the detection process. A velocity cone is computed similarly to a normal cone.

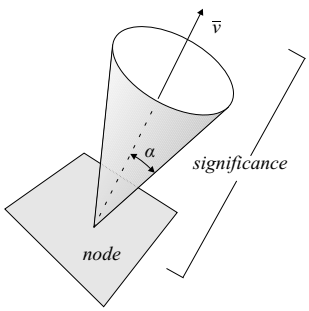



Figure 6: Velocity Cone.

It represents an approximation of the velocity distribution in a hierarchy node by a small number of values. On the one hand this permits fast calculation during the hierarchy update, and on the other hand the velocities of two nodes can efficiently be compared. The angle $\alpha$, the direction $\overline{v}$, and the height of the cone depend on the movement of the vertices. In particular, $\alpha$ measures the correlation of significant velocity vectors, and the height represents the total significance (e.g. the maximum velocity $\hat{v}$) of the movement.

## 6 Collision Detection and Distance Computation

We test two meshes for overlaps by recursively traversing the inflated hierarchies from top to down. Whenever two nodes overlap, all children inside the longer $k$-DOP are tested against the shorter one.

### 6.1 Proximity and Distance

Whenever two colliding hierarchy leaves have been found, the distance between each pair of faces is calculated, and candidate pairs are detected and passed to the collision response. To handle not only triangles but also polygonal primitives, we compute the closest points between convex polygons with an adapted implementation of the GJK algorithm [12].

We do not restrict the proximity detection to the simple particle–face test, since it is not sufficient for an accurate collision detection and limits cloth modelling to high-resolution meshes (figure 7).

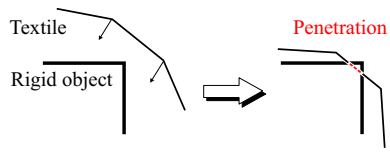



Figure 7: Particle based collision detection is inexact and resolution dependent.

Alternatively, virtual particles [11] can be inserted at critical positions, however they require additional costly calculations.

In order to handle multiple collisions that occur when textiles are clamped between other textiles or body parts, all critical proximities are passed to the collision response to ensure a smooth and accurate response.

### 6.2 Self-collision

We traverse the hierarchy of a deformable object by first checking whether the surface curvature criterion indicates proximities. In this case the child regions are recursively checked. Additionally, to detect proximities across the child borders, the child regions are recursively tested against each other similarly to the standard detection process.

The faces of two overlapping leaves are first tested for adjacency. If the faces belong to the same or to two adjacent groups (section 5.1), only non-adjacent faces with a significant angle are tested against each other, since contiguous faces on flat surfaces are not candidates for the collision response.

This method for self-collision detection turns out to be very efficient and only needs a fractional amount of the total time used for the collision detection.

### 6.3 Exploiting Coherence

A separation list as proposed by Li and Chen [20] can be built to detect frame-to-frame coherence and to reduce the costs for the hierarchy traversal. The list stores the node pairs where the last recursion stopped and the next detection process resumes the recursion at these nodes. Instead of checking whether a separation node moves up in the recursion tree, we just track the nodes moving down and rebuild the separation list after a while. The check for upwards moving nodes is expensive and usually fails anyway, as contacts in cloth simulation often persist for a longer period of time.

However, we found out that due to the large number of collisions occurring in cloth simulation, the maintenance of the separation list mostly takes more time than rerunning the $k$-DOP overlap tests.

Instead, in still scenes the velocity cones (section 5.2) are useful to detect nodes with small relative velocities, as for those nodes the detection results from the previous time step can be collected. The closest points of triangles are stored by their barycentric coordinates, thus they do not need to be recalculated during coherent movements. Assuming sufficient planarity of the faces, this is also valid for faces with more than three vertices. As errors may accumulate, the results have to be recomputed after a certain period of time depending on the velocities and the $\varepsilon_{close}$–distance analogically to the lazy hierarchy update (section 4.3).

## 7 Collision Response

For animated scenes an efficient collision response method is crucial. In particular, numerical stability and performance in the context of large time steps has to be preserved and no additional stiffness should be introduced into the system. In our cloth simulation system we therefore use constraints based on the efficient implementation presented by Baraff and Witkin [2].

To allow large time steps and preserve stability, our approach for collision detection and response aims at avoiding collisions before they occur. Therefore, not only colliding faces are detected but also proximities, that is faces closer than a distance $\varepsilon_{close}$. In this case, the collision detection returns pairs of colliding or close faces together with the closest points on these faces. While the collision detection process returns candidate faces, an adequate collision response has to be calculated for each involved particle. For that purpose, the collisions are sorted by the estimated distance of the two faces after the next time step. This distance is estimated by the current distance and the relative velocity.

Contrary to earlier approaches [15, 25], the velocity is not constrained in the normal direction of the close object but in direction of the closest point pair of the close faces (figure 8). There is no difference between these direction for collisions between smooth surfaces. However, for collisions between non-smooth surfaces and edges, this approach leads to a more stable collision response because there are no discontinuities in the constrained direction.

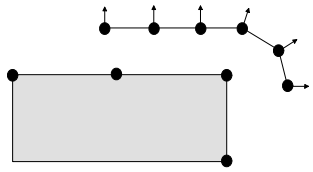


Figure 8: Constraint direction for different cloth particles at edge of rigid object.

If the velocity is constrained, it has to be preset with appropriate values. The collision response distinguishes between two different cases:

1. Collisions of two deformable faces.
2. Collisions between a deformable face and a face of the pre-computed rigid environment.

In both cases only faces moving towards each other are handled. Their velocity in the constrained direction is set to assert a minimum distance $d_{min}$.

This velocity is scaled by $\gamma = v_{rel}/\frac{d_{min}}{\Delta t}$ to simulate a resting contact, where $v_{rel}$ is the relative velocity of the two faces and $\Delta t$ is the current time step.

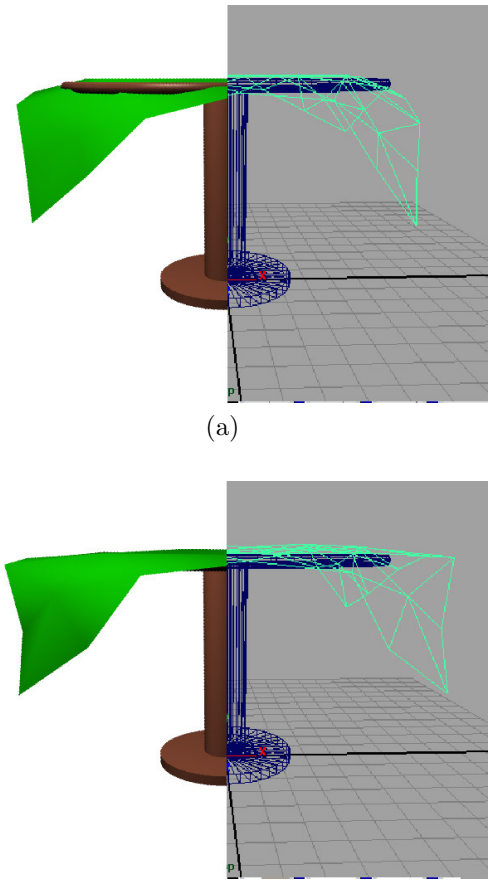
In the first case no velocity and therefore no momentum is transferred between the two faces of the object. In the second case additionally to the velocity used to assert the minimum distance, the velocity of the rigid object in the constrained direction has to be added to the velocity of the deformable object.

Thus, we constrain the velocity of the colliding particles in the direction $n_s$ of the closest point pair of the two faces and preset it to

$$v = \frac{1}{\Delta t}\gamma\left(d_{min} - d\right) + \pi\left(v_r\right) , \qquad (3)$$

where $\pi$ is the projection onto the oriented direction $n_s$, $d$ is the distance between the two faces and $v_r$ is the velocity of the rigid object. If interpenetrations occur, the faces are separated again, as long as their distance is smaller than a preset value.

Since we constrain only one direction of the particle velocity, the particle is still free to move according to the forces acting on it in the other directions. Hence it is easy to model stiction by constraining the other directions and replacing the projected velocity of the rigid object by the entire velocity of the rigid object in equation (3) such that the cloth moves with the rigid object. Collision constraints must be released when forces drag the particle away from the collision object. These release forces are estimated at the beginning of every integration step.



(a)

(b)

Figure 9: Accuracy of collision detection and response in Maya Cloth (a) compared to our system (b).

# 8 Results

We demonstrate first the accuracy and second the performance of our system. Afterwards, the results are summarized.

## 8.1 Accuracy

Several professional cloth modelling systems are available for purchase. We compare the accuracy of our system with "Cloth" included in Maya® 4 Unlimited[1]. Figure (9) shows the scene "tableCloth" consisting of a low-resolution table cloth (49 vertices, 72 triangles), which drapes over a round table. Both "Cloth" and our system compute the simulation of the falling cloth in real-time, but "Cloth" only tests vertices with the collision object and produces visually poor results due to penetrations with the edge of the table. Our system correctly detects all proximities and the constraints safely prevent intersections.

## 8.2 Performance

In order to show the performance of the new methods, we compare the methods with four different test scenes (table 1). The columns list the number of particles, the number of polygons of the unde-

[1] *Maya*® by Alias|Wavefront

formable objects and the computation times needed for the simulation. In particular, these are the duration of the collision detection (CD), the collision response (CR), the numerical solution of the particle system and finally the total simulation, not regarding the output. While the pants in the "Walk pants" scene are simulated as triangle-meshes, the textiles of the other three tests are quad-meshes. The rigid objects are all stored as triangle-meshes.

In the walking scenes a male avatar makes six steps within about six seconds. One test is performed wearing some pants that were created according to real garments (scene "Walk pants", figure 12). The other two "Walk" scenes (figure 13) show a sweater and slacks that are present in two different resolutions, namely a very low-resolution (LR) and a very high-resolution version (HR). Thus, the dependency on the complexity of the meshes can be analyzed and the robustness of collision detection and response can be compared.

Finally, the scene "Tape" (figure 14) consists of a tape of one meter in length, which is falling onto a solid object and comes to rest after four seconds. This scene particularly challenges the accuracy of the self-collision detection and response.

All benchmarks are computed on a PC with a Pentium 4 processor running at 2 GHz, and the simulations are calculated with constant time steps of size $0.01s$. As well as the benchmarks also the images are calculated with this step size, they are not generated using smaller step sizes to get possibly better visual results.

The collision detection defaults to 18-DOPs, quad-trees and oriented inflation. If simpler bounding volumes are used, the detection speed decreases considerably. Tables (2, 3, 4, 5) list some examples for non-optimal values, where "HU" and "CT" denote the computation times for the hierarchy update and the intrinsic collision test respectively. For the unoriented inflation an offset of $\varepsilon_{close} = 2cm$ was used to insure robust detection and response.

## 8.3 Summary

Evidently the collision detection strongly improves accuracy and performance by the advances described in this paper. The oriented inflation allows the implicit solver to choose large time step sizes. Common bounding volumes have to be in-

| Collision detection setup | HU | CT | Total |
|---|---|---|---|
| Optimal | 94 | 52 | 147 |
| Unoriented instead of oriented inflation | 62 | 227 | 288 |
| AABBs instead of 18-DOPs | 72 | 83 | 155 |
| Binary trees instead of quadtrees | 107 | 58 | 165 |

Table 2: Collision detection times for scene "Walk pants" measured in $ms$ per frame.

| Collision detection setup | HU | CT | Total |
|---|---|---|---|
| Optimal | 59 | 47 | 106 |
| Unoriented instead of oriented inflation | 58 | 133 | 192 |
| AABBs instead of 18-DOPs | 78 | 104 | 182 |
| Binary trees instead of quadtrees | 93 | 49 | 142 |

Table 3: Collision detection details for scene "Walk LR" measured in $ms$ per frame.

| Collision detection setup | HU | CT | Total |
|---|---|---|---|
| Optimal | 114 | 49 | 163 |
| Unoriented instead of oriented inflation | 77 | 289 | 367 |
| AABBs instead of 18-DOPs | 101 | 100 | 201 |
| Binary trees instead of quadtrees | 133 | 50 | 183 |

Table 4: Collision detection details for scene "Walk HR" measured in $ms$ per frame.

| Collision detection setup | HU | CT | Total |
|---|---|---|---|
| Optimal | 10 | 47 | 57 |
| Unoriented instead of oriented inflation | 9.6 | 343 | 354 |
| AABBs instead of 18-DOPs | 6.5 | 72 | 78 |
| Binary trees instead of quadtrees | 9.5 | 51 | 61 |

Table 5: Collision detection details for scene "Tape" measured in $ms$ per frame.

| Scene | Particles | Solid polygons | CD | CR | Solver | Total |
|---|---|---|---|---|---|---|
| Walk pants | 833 | 28784 | 88$s$ | 24$s$ | 35$s$ | 147$s$ |
| Walk LR | 713 | 28784 | 63$s$ | 25$s$ | 89$s$ | 177$s$ |
| Walk HR | 10757 | 28784 | 98$s$ | 41$s$ | 6430$s$ | 6570$s$ |
| Tape | 1449 | 580 | 23$s$ | 7.0$s$ | 315$s$ | 347$s$ |

Table 1: Scenes and computation times for the benchmarks.

tensively inflated in order to achieve an accurate simulation and result in a severe performance loss for the hierarchies. Furthermore, $k$-DOPs approximate the textiles much better than simple axis aligned bounding boxes and provide a reasonable speed-up. A comparable speed-up is additionally achieved by the higher order hierarchy trees.

Further details about some of the methods and results can be found in [23].

# 9 Conclusions

In this work we have shown that the notion of object hierarchies for collision detection for cloth models can be advanced by an intelligent choice of methods for all components of the detection, namely hierarchy building, update, and traversal. Moreover, an extended set of heuristics improves the performance further such that the collision detection is no longer a bottle neck in cloth modeling systems.

More precisely we showed

- that **$k$-DOPs** are well suited for collision detection between deformable and flat shaped meshes like textiles
- how $k$-DOP hierarchies can be extended to **proximity detection** with acceptable overhead
- that it is worth while considering **other trees** than binary trees if the bounding volume overlap test is fast
- how **normal cones** can be incorporated into $k$-DOP hierarchies and how the concept of **face groups** can still guarantee a correct self-collision detection
- a way to easily represent movements of hierarchy nodes using **velocity cones**
- that **constraints** keep the collision response stable even for low-resolution meshes needed in **real-time applications**.

Future work will include the development of an application of the presented hierarchies for multi-resolution models.

# References


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

[2] D. Baraff and A. Witkin. Large Steps in Cloth Simulation. *Computer Graphics*, 32(Annual Conference Series):43–54, 1998.

[3] G. Barequet, B. Chazelle, L. J. Guibas, J. S. B. Mitchell, and A. Tal. BOXTREE: A Hierarchical Representation for Surfaces in 3D. *Computer Graphics Forum*, 15(3):387–396, 1996.

[4] R. Bigliani and J. W. Eischen. Collision Detection in Cloth Modeling. In *Cloth and Clothing in Computer Graphics*. ACM SIGGRAPH, 1999.

[5] 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, 218, 1995.

[6] M. Courshesnes, P. Volino, and N. Magnenat-Thalmann. Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 137–144. ACM SIGGRAPH, Addison Wesley, August 1995.

[7] M. Desbrun and M.-P. Cani-Gascuel. Active implicit surface for animation. In *Graphics Interface*, pages 143–150, 1998.

[8] B. Eberhardt, J.-U. Hahn, R. Klein, W. Straßer, and A. Weber. Dynamic Implicit Surfaces for Fast Proximity Queries in Physically Based Modeling. Technical Report WSI-2000-11, Eberhard-Karls-Universität Tübingen, June 2000.

[9] S. A. Ehmann. SWIFT - Speedy Walking via Improved Feature Testing. http://www.cs.unc.edu/∼geom/SWIFT/.

[10] S. A. Ehmann and M. C. Lin. Accurate and Fast Proximity Queries Between Polyhedra Using Surface Decomposition. In *Computer Graphics Forum (Proc. of Eurographics)*, 2001.

[11] O. Etzmuss, B. Eberhardt, M. Hauth, and W. Strasser. Collision Adaptive Particle Systems. *Proc. of Pacific Graphics*, 2000.

[12] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space. *IEEE Journal of Robotics and Automation*, 4(2), 1988.

[13] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A Hierarchical Structure for Rapid Interference Detection. *Computer Graphics*, 30(Annual Conference Series):171–180, 1996.

[14] A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. *Proc. ACM SIGMOD Conference, Boston*, pages 47–57, 1984.

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

[16] T. He. Fast Collision Detection Using QuOSPO Trees. *Proc. of the 1999 symposium on Interactive 3D graphics*, pages 55–62, 1999.

[17] P. M. Hubbard. Approximating Polyhedra with Spheres for Time-Critical Collision Detection. *ACM Transactions on Graphics*, 15(3):179–210, 1996.

[18] D. Johnson and E. Cohen. Spatialized Normal Cone Hierarchies. In *ACM Symposium on Interactive 3D Graphics*. ACM SIGGRAPH, 2001.

[19] J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral, and K. Zikan. Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998.

[20] T.-Y. Li and J.-S. Chen. Incremental 3D Collision Detection with Hierarchical Data Structures. In *Proc. of the ACM Symposium on Virtual reality software and technology*, 1998.

[21] M. C. Lin and J. F. Canny. A Fast Algorithm for Incremental Distance Calculation. In *IEEE International Conference on Robotics and Automation*, pages 1008–1014, 1991.

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

[23] Johannes Mezger. Effiziente Kollisionsdetektion in der Simulation von Textilien, 2001. Diploma Thesis, WSI/GRIS, Universität Tübingen.

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

[25] X. Provot. Collision and Self-Collision Handling in Cloth Model Dedicated to Design Garments. In *Graphics Interface*, pages 177–189. Canadian Information Processing Society, Canadian Human-Computer Communications Society, May 1997.

[26] G. van den Bergen. Efficient Collision Detection of Complex Deformable Models using AABB Trees. *Journal of Graphics Tools*, 2(4):1–14, 1999.

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

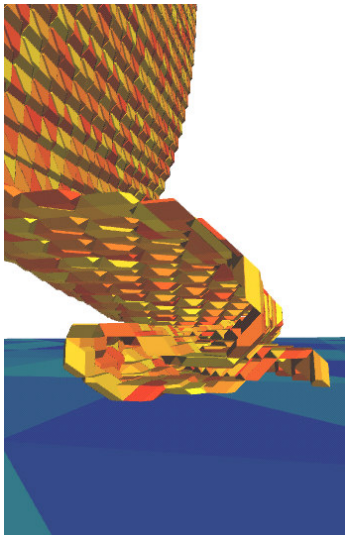
[28] P. Volino and N. Magnenat-Thalmann. Efficient Self-Collision Detection on Smoothly Discretized Surface Animations using Geometrical Shape Regularity. *Computer Graphics Forum*, 13(3):155–166, 1994.

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

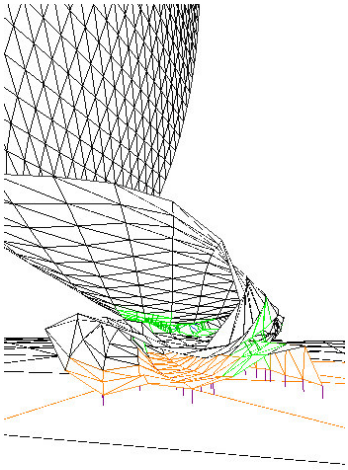
[30] P. Volino and N. Magnenat Thalmann. Accurate Collision Response on Polygonal Meshes. *Computer Animation Conference*, 2000.

[31] G. Zachmann. Rapid Collision Detection by Dynamically Aligned DOP-Trees. *Proc. of IEEE, VRAIS'98 Atlanta*, 1998.

[32] D. Zhang and M. M.F. Yuen. Collision Detection for Clothed Human Animation. *Proc. of Pacific Graphics*, 2000.



(a)



(b)

Figure 10: Inflated 18-DOPs (a) and detected closest points (b) of a tape draping onto a plane.

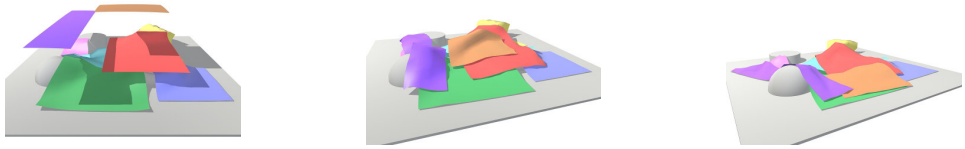


Figure 11: Sheets of cloth falling on geometric objects (441 particles per sheet).

Figure 12: Walking man (833 particles for clothing, 28784 particles for avatar).

Figure 13: Walking man (10757 particles for clothing, 28784 particles for avatar).

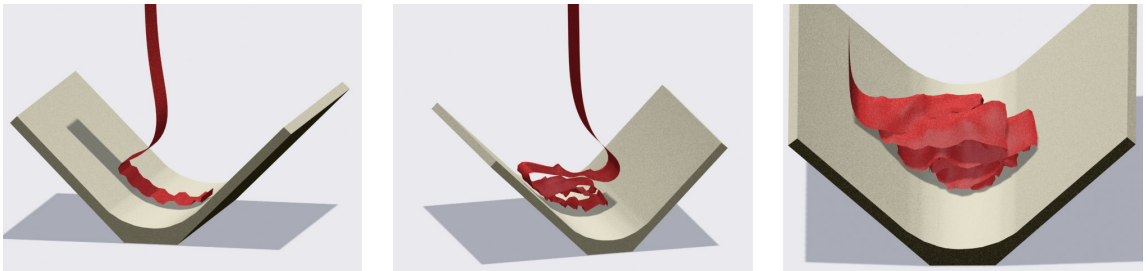


Figure 14: Falling tape with 1449 particles.