Hierarchical Techniques in Collision Detection for Cloth Animation

J. Mezger, S. Kimmerle, O. Etzmuß,
WSI/GRIS
University of Tübingen
Sand 14,
D-72076 Tübingen, Germany
{mezger, kimmerle, etzmuss}@gris.uni-tuebingen.de

ABSTRACT

In the animation of deformable objects, collision detection is 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 pruning of the hierarchy. Oriented inflation of bounding volumes enables us to detect proximities with a minimum of extra cost.

Keywords
Computer Graphics, Cloth Simulation, Collision Detection.

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.

In this paper we employ the notion of object-based hierarchies, first applied to cloth modelling by Volino et al. [VMT94]. The hierarchical representations of all objects including the deformable surfaces of arbitrary meshed textiles are built in a pre-processing 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.

2. PREVIOUS WORK

Many collision detection methods for various purposes have been developed in the past [LG98]. 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

Journal of WSCG, Vol.11, No.1., ISSN 1213-6972
WSCG 2003, February 3-7, 2003, Plzen, Czech Republic.

[GJK88], Lin-Canny-Algorithm [LC91] or V-Clip [Mir98]. Non-convex objects can be decomposed into convex parts [EL01; Ehm]. R-trees [Gut84] provide the theoretical basics for bounding volume hierarchies, which are mostly used to generate hierarchical representations of complex meshes. In addition, possibly colliding objects are identified by Sweep-and-Prune strategies [CLMP95]. As opposed to bounding volume hierarchies, regular grids partition the scene into voxels [BE99; ZY00]. Alternatively, graphics hardware [BWS98] can be employed to detect collisions in image-space, which was even investigated for cloth modelling [VSC01].

Particular advances in accelerating the self-collision detection are achieved by Volino et al. [VMT94]. 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 [CVMT95]. Recent publications [VMT00] additionally address k-DOPs as bounding volumes. Provot [Pro97] describes a similar approach for the surface curvature heuristic, which we extend in our system. Johnson et al. [JC01] show how normal cone hierarchies can accelerate not only distance computations, but also lighting and shadowing.

3. BOUNDING VOLUMES

In complex dynamic scenes, bounding volumes have to be permanently readapted to the approximated geometry. For this application we choose a bounding volume hierarchy of k-DOPs [KHM+98].

The advantages of this choice over other hierarchies are identified in section 4.

3.1 k-DOPs

A k-DOP [KHM+98] (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 components 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 $N_6 = \{(\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1)\}$, an octahedron (8-DOP) is generated by setting all normal components to ± 1. We usually use 14-DOPs ($N_{14} = N_6 \dot{\cup} N_8$) or 18-DOPs (AABB with clipped edges).

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

3.2 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 ε_{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 can easily be verified that the overlap of such two enlarged k-DOPs is a necessary condition for actual ε_{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 adapted to enclose 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).

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

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

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

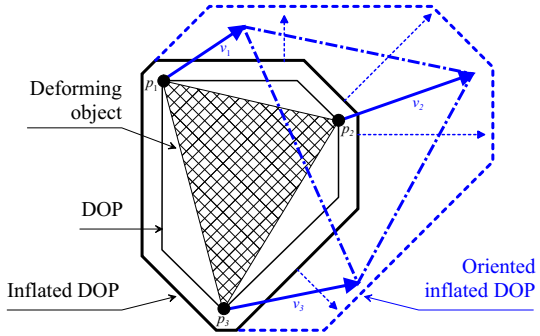


Figure 1: Estimated movement and oriented inflation of the 8-DOP.

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 [ZY00], 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 [VSC01] 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.

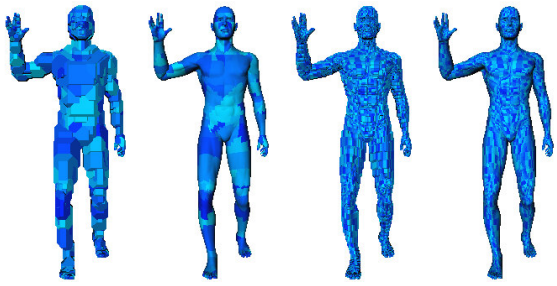
Therefore, a realistic cloth modelling system requires bounding volume hierarchies to be robust and efficient at the same time. We propose to combine the advantages of a top-down k-DOP hierarchy with a surface curvature criterion.

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, as for AABBs, k-DOP bounding volumes satisfy the equation

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

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 [VMT94], 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 the dynamic addition of objects 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).

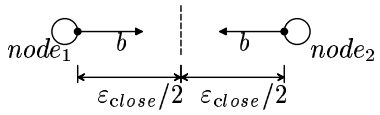


Figure 3: Tolerance distance for the lazy hierarchy update.

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.

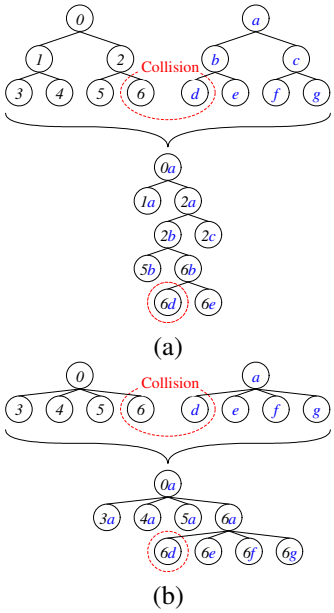


Figure 4: Recursion using binary trees (a) and quadtrees respectively (b).

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. In particular, they are significantly faster than binary trees.

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

An exact method to reject possible self-intersections for a certain region was suggested by Volino and Magnenat-Thalmann [VMT94], 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 [Pro97], 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 α 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.

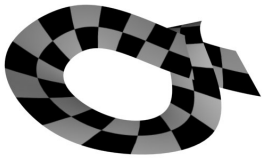


Figure 5: Self-intersecting mesh with correlated normals but concave shape.

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.

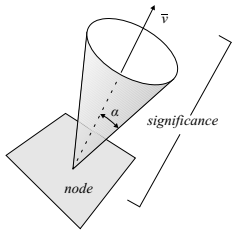


Figure 6: Velocity Cone.

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 α, the direction $\overline{v}$, and the height of the cone depend on the movement of the vertices. In particular, α 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 the top 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 [GJK88].

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).

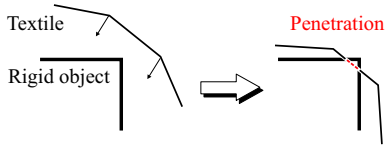


Figure 7: Particle based collision detection is inexact and resolution dependent.

Alternatively, virtual particles [EEHS00] 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 [LC98] 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

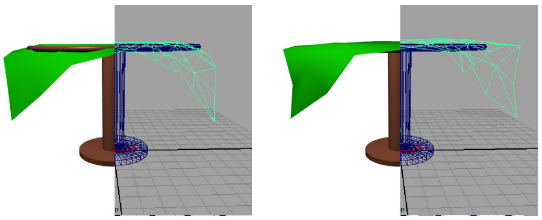
expensive and usually fails anyway, as contacts in cloth simulation often persist for a longer period of time.

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 ε_{close}–distance analogically to the lazy hierarchy update (section 4.3).

7. RESULTS

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 (8) 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.



(a) (b)

Figure 8: Accuracy of collision detection and response in Maya Cloth (a) compared to our system (b).

Another test is performed on a walking avatar (28784 polygons). The avatar is dressed with pants assembled of several garment patterns (833 vertices, 1626 triangles). In table (1) the collision detection performance is compared for a walk over 6 seconds. The result for our accelerated configuration (18-k-DOPs, quadtrees) is listed in the first row. The other rows show the results that are obtained when simpler collision detection methods are used. For the unoriented inflation an offset of $\varepsilon_{close} = 2cm$ was used to insure robust detection and response. The columns list the run-times of the hierarchy update (HU), collision test (CT), and the total time spent on the collision detection.

Collision detection setup	HU	CT	Total
Accelerated configuration	56	31	88
Unoriented instead of oriented inflation	37	136	173
AABBs instead of 18-DOPs	43	50	93
Binary trees instead of quadtrees	64	35	99

Table 1: Collision detection times for the walking avatar measured in seconds for the simulation of 600 time steps.

For the simulation, the collision response took 24 seconds, the numerical solution of the particle system 35 seconds. In our simulations we applied the collision response scheme described in [MKE02]. The time step size was set to $0.01s$ for both collision detection and the solver. Thus, 600 time steps had to be computed overall. Figure (9) shows some pictures from the simulation.

As a result of the oriented inflation, proximities between several moving textiles are accurately detected. Figures (10) and (11) show examples for complex collisions and self-collisions with high relative velocities.

Evidently, the collision detection performance is strongly improved 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 intensively 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.

8. 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 further improves the performance 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

[1] *Maya*® by Alias|Wavefront

- 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**.

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

9. REFERENCES

[BE99] R. Bigliani and J. W. Eischen. Collision Detection in Cloth Modeling. In *Cloth and Clothing in Computer Graphics*. ACM SIGGRAPH, 1999.

[BWS98] G. Baciu, W. Wong, and H. Sun. Hardware-Assisted Virtual Collisions. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST, Taipei, Taiwan*, pages 145–151, 1998.

[CLMP95] 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.

[CVMT95] 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.

[EEHS00] O. Etzmuss, B. Eberhardt, M. Hauth, and W. Strasser. Collision Adaptive Particle Systems. *Proceedings of Pacific Graphics*, 2000.

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

[EL01] 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.

[GJK88] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space. *IEEE Journal of Robotics and Automation*, 4(2), 1988.

[Gut84] A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. *Proc. ACM SIGMOD Conference, Boston*, pages 47–57, 1984.

[JC01] D. Johnson and E. Cohen. Spatialized Normal Cone Hierarchies. In *ACM Symposium on Interactive 3D Graphics*. ACM SIGGRAPH, 2001.

[KHM+98] J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral, and K. Zikan. Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998.

[LC91] 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.

[LC98] T.-Y. Li and J.-S. Chen. Incremental 3D Collision Detection with Hierarchical Data Structures. In *Proceedings of the ACM Symposium on Virtual reality software and technology*, 1998.

[LG98] M. C. Lin and S. Gottschalk. Collision Detection Between Geometric Models: A Survey. *Proc. of IMA Conference on Mathematics of Surfaces*, 1998.

[Mir98] B. Mirtich. VClip: Fast and Robust Polyhedral Collision Detection. *ACM Transactions on Graphics*, 17(3):177–208, 1998.

[MKE02] J. Mezger, S. Kimmerle, and O. Etzmuß. Improved Collision Detection and Response Techniques for Cloth Animation. Technical report, Universität Tübingen, 2002.

[Pro97] X. Provot. Collision and Self-Collision Handling in Cloth Model Dedicated to Design Garments. In *Graphics Interface '97*, pages 177–189. Canadian Information Processing Society, Canadian Human-Computer Communications Society, May 1997.

[VMT94] 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.

[VMT00] P. Volino and N. Magnenat-Thalmann. Implementing fast Cloth Simulation with Collision Response. In *Computer Graphics International*, June 2000.

[VSC01] T. Vassilev, B. Spanlang, and Y. Chrysanthou. Fast Cloth Animation on Walking Avatars. In *Computer Graphics Forum (Proc. of Eurographics)*, 2001.

[ZY00] D. Zhang and M. M.F. Yuen. Collision Detection for Clothed Human Animation. *Proceedings of the Pacific Graphics*, 2000.

(a) (b) (c)

Figure 9: Walk with 833 particles for trousers, 28784 vertices for avatar.
(a, b): Pictures from the simulation in Table 1. (c): Another simulation with shirt (1138 particles).

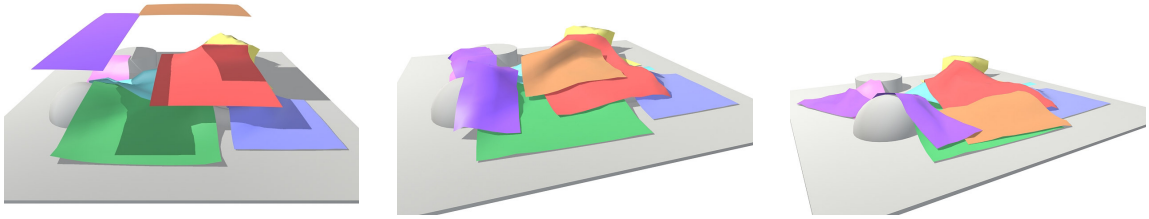


Figure 10: Sheets of cloth falling on geometric objects (441 particles per sheet).

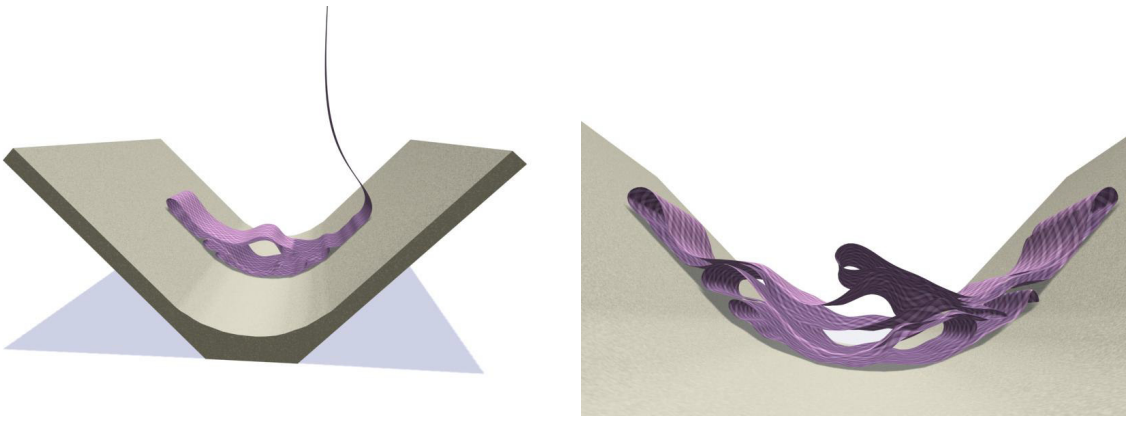


Figure 11: Dropping a long tape on a curved surface (1449 particles).