Optimized Continuous Collision Detection for Deformable Triangle Meshes

Marco Hutter
Fraunhofer IGD, Darmstadt, Germany
Marco.Hutter@igd.fraunhofer.de

Arnulph Fuhrmann
Fraunhofer IGD, Darmstadt, Germany
Arnulph.Fuhrmann@igd.fraunhofer.de

ABSTRACT

We present different approaches for accelerating the process of continuous collision detection for deformable triangle meshes. The main focus is upon the collision detection for simulated virtual clothing, especially for situations involving a high number of contact points between the triangle meshes, such as multi-layered garments. We show how the culling efficiency of bounding volume hierarchies may be increased by introducing additional bounding volumes for edges and vertices of the triangle mesh. We present optimized formulas for computing the time of collision for these primitives analytically, and describe an efficient iterative scheme that ensures that all collisions are treated in the correct chronological order.

Keywords: collision detection, cloth simulation, continuous collision detection

1 INTRODUCTION

In physically based simulation, collision detection is essential for a realistic behavior of the simulated objects. Efficient solutions for collision detection of rigid bodies have been developed, but in the simulation of deformable objects, collision detection is still the bottleneck. Especially for two-manifold surfaces like cloth, there arise some major difficulties: These objects are infinitely thin, and even small interpenetrations, for example, between two layers of simulated cloth, may cause visually distracting artifacts. Modeling a realistic cloth thickness makes it necessary to check the objects for close proximity rather than for contact, but still this technique limits the size of the simulation time step, or analogously, the maximum velocity of the objects. One solution for this problem is continuous collision detection. It allows all collisions and proximities to be detected even for large simulation time steps. Of course, this robustness may only be achieved with higher computational cost.

One major objective of this paper is to show how robust continuous collision detection can be made more efficient, in order to employ it for the simulation of cloth and multilayered virtual garments. We show how the number of collision tests between triangle meshes in close proximity can be significantly reduced by introducing additional bounding volumes for the primitives. We also present optimized formulas for continuous collision tests. Another contribution is a method for increasing the speed of the iterative collision detection, which marks the parts of the triangle mesh in which further collision tests have to be performed, and thus allows us to skip large areas where all collisions have already been resolved.

2 RELATED WORK

The problem of collision detection has recently been addressed by many authors, since its importance for physically based simulation has become obvious [12]. Common strategies for accelerating the process of collision detection have been studied in detail, especially the use of bounding volume hierarchies [11, 10, 28], and the application of different types of bounding volumes, such as bounding spheres [13], oriented bounding boxes (OBB) [4], axis-aligned bounding boxes (AABB) [20], polytopes with k discrete face orientations (k-DOP) [9] and combinations of these [15]. Alternative approaches, like spatial hashing [19], voronoi diagrams [17] or GPU-accelerated hierarchical techniques [6] have also been examined. In order to accelerate the process of collision detection, stochastic methods have also been applied [8, 27].

For the challenging field of collision detection of deformable objects like cloth, different approaches have been suggested. These approaches deal with the optimization of bounding volume hierarchies for deformable objects, and the exploitation of special geometric properties of such objects, in order to increase the efficiency of self-collision detection. These properties include curvature criteria [23, 14, 26] or special adjacency information for triangle meshes [5]. Some authors described methods for correcting invalid simulation states that occur due to non-robust collision detection [22, 1, 25], whereas others focused on the robust continuous collision detection, which preserves an intersection-free state throughout the whole simulation [2, 7]. A recent comprehensive survey on collision

detection for deformable objects can be found in the paper *Collision Detection for Deformable Objects* [18].

3 OVERVIEW

In Section 4, we give a short outline of the simulation process and show how the collision detection is applied during the simulation cycle. Additionally, we introduce some basic concepts for collision detection, and describe which problems occur when these concepts are employed for the collision detection of multilayered garments.

In Section 5, we show how these concepts may be improved in order to increase the efficiency for situations involving multiple layers of triangle meshes in close proximity.

Optimizations for continuous collision detection are presented in Section 6. These include optimized formulas for a single continuous collision test, as well as optimizations for the iterative scheme that is applied for continuous collision detection.

Section 7 summarizes the results of the presented approaches for several test cases, and Section 8 contains the conclusions and a short outlook on future developments.

4 SIMULATION AND COLLISION DETECTION

4.1 Simulation

We use a simulation scheme similar to that proposed by Bridson et al. [2], who emphasized that the collision detection and response should be applied to a *candidate* state that is obtained from the simulator, in order to cleanly separate the simulation process from the collision detection. Given a state consisting of particle positions and velocities of the simulated particle system $(\mathbf{x}^n, \mathbf{v}^n)$ at time t_n, the candidate velocities $\mathbf{v}^{n+\frac{1}{2}}$ for the next time step are computed by the simulator. Then collision detection is applied, and the collision responses solely affect the candidate state: The candidate velocities are modified in order to prevent collisions or interpenetration. Finally, a velocity $\mathbf{v}^{n+1}$ is obtained, which describes a collision-free movement, and the particle positions for the next time step may be computed as $\mathbf{x}^{n+1} = \mathbf{v}^{n+1} \cdot \Delta t$.

4.2 Collision Detection

The task of checking two static objects for collisions may by accomplished by checking each pair of primitives if they have a distance that is smaller than a certain threshold. This process has an inherent worst-case complexity of $O(n^2)$, but since this worst case hardly ever occurs in a realistic scenario, the process of collision detection is usually divided into two phases: In the *broad phase*, conservative tests are performed in order to cull away many pairs of primitives that may not collide. In the *narrow phase*, exact tests between relatively few pairs of primitives are performed. It is sufficient to perform the collision tests for pairs consisting of a vertex and a triangle, or two edges. Robust and efficient techniques for computing the closest points between primitives are, for example, described in the book *Geometric Tools for Computer Graphics* [16]. If the distance between these primitives is too small, a collision response is applied in order to maintain the distance that is given by the thickness of the simulated material. This collision response may consist of a stiff spring or a collision impulse that is applied between the closest points of the primitives.

Bounding Volume Hierarchies (BVH)

The most common and efficient approach for the broad phase of the collision detection are bounding volume hierarchies (BVH). The basic idea is to hierarchically divide a set of primitives into subsets, where the leaves of a BVH usually contain single triangles. The subsets of primitives are approximated with bounding volumes. The bounding volumes may be 'blown up' by the thickness of the simulated material, in order to detect not only intersections, but also close proximities between pairs of primitives. Traversing the hierarchy and performing overlap tests between the bounding volumes allows us to cull away many pairs of primitives that may not collide, exploiting the fact that collisions between the primitives of two sets may only occur if their bounding volumes overlap. Usually there is a trade-off between the quality of the approximation of the underlying geometry, and the cost for the creation of the bounding volume and single intersection test. For deformable objects, the BVH has to be refitted according to the deformation the geometry is undergoing in each time step, and in these cases the cost for updating the bounding volumes also has to be taken into account. Despite the high cost for intersection tests and updating, our tests have shown that k-DOPs are the most efficient choice for tasks like cloth simulation, because the tighter approximation of the (typically heavily non-convex) geometry reduces the number of *false positives* in the broad phase, and thus, fewer exact collision tests have to be performed.

Two major problems occur when BVHs are employed for the collision detection of deformable triangle meshes:

- **A high number of false positives during self-collision detection.**

  Self-collision detection is usually performed by clipping a BVH against itself. Due to the fact that the bounding volumes of two adjacent triangles always overlap, the traversal often reaches pairs of leaf nodes only because the triangles contained in

these nodes are adjacent. The methods suggested by Volino and Magnenat-Thalmann [23], Provot [14], Wong and Baciu [26] and Govindaraju et al. [5] alleviated these problems and increased the efficiency of self-collision detection, so that the second problem drew more attention:

- **A high number of false positives between multiple layers**

  Collision detection between different objects that are given as deformable triangle meshes is performed by clipping the BVH of the respective objects against each other. When the meshes are similarly triangulated, and form multiple layers in close proximity, the bounding volume of each leaf node of one mesh on average intersects six bounding volumes of triangles of each adjacent layer.

4.3 Continuous Collision Detection

For the simulation with deformable 2-manifold triangle meshes, the collision tests are usually performed in each time step. But checking for proximity does not prevent two objects passing through each other in a single time step. In order to detect *all* collisions, continuous collision tests have to be performed. These will allow us to prevent interpenetration of the objects regardless of the size of the time step, the velocity of the objects, and the thickness of the material. For each pair of primitives, one has to detect if they will collide in the current time step, and at which time this collision will occur.

Two problems arise when continuous collision detection is used:

- Collisions that occur *later* in time may be detected *earlier* in the collision detection process. For an accurate collision response and for a plausible behavior of the simulated objects, the collisions should be treated in the correct chronological order.

- Collision responses may cause new, secondary collisions. These collisions may cause interpenetration, if they are not detected and resolved.

In order to alleviate the problem of secondary collisions caused by the responses to preceding collisions, some authors (e.g. [21, 2]) suggested to use an iterative process. A method for treating the collisions in the correct chronological order was suggested by Eberle [7]. For each triangle, only the earliest collision that occured should be taken into account. Combining these techniques allows a robust and plausible treatment of all collisions that occur in one time step.

5 BOUNDING VOLUMES FOR PRIMITIVES

When the BVH is traversed and the bounding volumes of two triangles overlap, for each vertex of one triangle the closest point on the other triangle has to be computed, yielding 6 point-triangle collision tests. Additionally, the closest points between each edge of one triangle and each edge of the other triangle have to be computed, yielding 9 edge-edge collision tests. As mentioned above, the bounding volume of each triangle of one mesh on average intersects six bounding volumes of triangles of a layer in close proximity. Without further precautions, this would cause $6 \cdot (6$ particle-triangle-tests + 9 edge-edge-tests), yielding 90 exact collision tests between pairs of primitives.

Since on average each edge is contained in two triangles, and each particle is contained in six triangles, some of these collision tests are redundant. In order to avoid these redundant tests, we store the information about the collision tests that have already been performed in the primitives they involve, similar to the method described by Wong and Baciu [26]. But still, for each triangle of one layer that lies flat on another layer, approximately 60 collision tests are performed.

In order to further reduce the number of collision tests that are caused by the false positives of the BVH, we propose introducing additional bounding volumes for the edges and vertices of the triangle meshes. These primitive bounding volumes may directly be stored in the edges and vertices, whereas the bounding volumes of the triangles are equal to the bounding volumes of the leaf nodes of the BVH, as depicted in Figure 1.

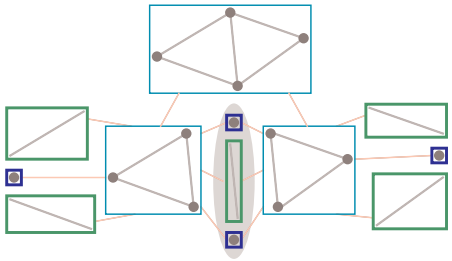

Figure 1: Additional bounding volumes for primitives: For two triangles of a triangle mesh, the bounding volumes are shown. The additional bounding volumes for the vertices and edges are shown with bold lines. The primitives that are shared among the two triangles are shaded.

For our implementation, we used 18-DOPs as bounding volumes for the primitives and the BVH. In each simulation step, the bounding volumes of the vertices are updated according to their current positions. The bounding volumes for the edges and triangles are updated by merging the bounding volumes of their vertices. Finally, the bounding volumes of the inner nodes

of the BVH are updated bottom-up, by merging the bounding volumes of their child nodes.

Although the bounding volumes for the vertices in this case only contain a single point, this yields a small advantage in the further update process: Combining a k-DOP with the pre-computed k-DOP of the vertex is in every case (with $k > 6$) computationally *cheaper* than combining the k-DOP with a single point describing the vertex position. Additionally, for the continuous collision detection discussed later, the vertex bounding volumes have to be extended so that they contain the positions of the vertices at the beginning and at the end of the time step. In this case the advantage of the vertex bounding volumes becomes even more obvious.

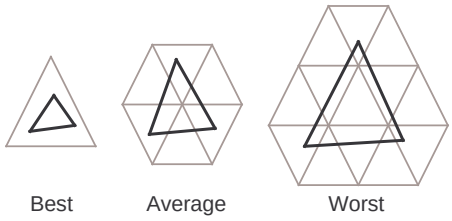

Figure 2: Different cases of triangles in close proximity

Case	Total	NR	PBV	NR+PBV	Coll.
Best	15	15	3	3	3
Avg.	90	61	23	11	10
Worst	195	123	102	54	54

Table 1: Number of exact collision tests for the cases depicted in Figure 2

- Total : Implied by the overlapping triangle bounding volumes
- NR : When no redundant tests are performed
- PBV : When primitive bounding volumes are used
- Coll : Number of collisions that actually occurred

Table 1 shows the number of collision tests performed for a single triangle, lying flat on a piece of another triangle mesh, for the different cases depicted in Figure 2. Of course, in the worst case one triangle may span an arbitrarily high number of triangles in another layer, but for similar triangulations, it should hardly span more than 13 triangles, as shown in the figure.

Approximately one third of the collision tests implied by the overlapping bounding volumes of triangles are redundant, and may easily be avoided. The use of primitive bounding volumes additionally decreases the number of false positives. Note that the exact number of collision tests that may be saved depends on the type of the bounding volumes. For 18-DOPs, the number of tests also depends on the orientation of the edges relative to the 18 directions of the DOP.

Since the number of edges is large for usual cloth triangle meshes (#edges = #vertices + #triangles - 1), some time for the update of the bounding volumes may be saved when the bounding volumes for the edges are only updated *on demand*. That means that the bounding volumes for edges are only updated when the bounding volumes of two triangles containing these edges overlap, and the bounding volumes of these edges have not yet been updated in the current time step.

6 EFFICIENT CONTINUOUS COLLISION DETECTION

Preventing all intersections between triangle meshes is very time-consuming. The primitive bounding volumes described in the previous section may also be applied for continuous collision detection, and dramatically reduce the number of collision tests that have to be performed. But still, the collision tests themselves and the iterative procedure allow further optimizations, which we will describe in the following sections.

6.1 Continuous Collision Tests

Exact collision tests are usually performed between vertices and triangles, and between pairs of edges. Both cases have in common that they involve four vertices. In each case, a necessary condition for a collision is, that the involved vertices are coplanar. Provot [14] showed that for four points with positions $\mathbf{x}_1 ... \mathbf{x}_4$ and constant velocities $\mathbf{v}_1 ... \mathbf{v}_4$, with $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ and $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$, the times t_i when the points are coplanar are the solutions of the equation

$$(\mathbf{x}_{21} + t \cdot \mathbf{v}_{21}) \times (\mathbf{x}_{31} + t \cdot \mathbf{v}_{31}) \cdot (\mathbf{x}_{41} + t \cdot \mathbf{v}_{41}) = 0 \quad (1)$$

For the algebraic solution, the monomial form of the polynomial is required. Computing the coefficients for this polynomial by simply expanding the differences, dot- and cross-products, and grouping the resulting terms by powers of t yields expressions that involve 188 additions, 192 multiplications, and are far from optimal (see [7]).

The coefficients may be computed with only 50 additions and 48 multiplications, by grouping equal terms and rewriting the coefficients as dot- and cross-products.

$$\begin{aligned} a_3 &= \mathbf{v}_{21} \cdot \mathbf{v}_{31} \times \mathbf{v}_{41} \\ a_2 &= \mathbf{x}_{21} \cdot \mathbf{v}_{31} \times \mathbf{v}_{41} - \mathbf{v}_{41} \cdot \mathbf{x}_{31} \times \mathbf{v}_{21} - \mathbf{v}_{21} \cdot \mathbf{x}_{41} \times \mathbf{v}_{31} \\ a_1 &= \mathbf{v}_{41} \cdot \mathbf{x}_{21} \times \mathbf{x}_{31} - \mathbf{x}_{21} \cdot \mathbf{x}_{41} \times \mathbf{v}_{31} - \mathbf{x}_{41} \cdot \mathbf{x}_{31} \times \mathbf{v}_{21} \\ a_0 &= \mathbf{x}_{41} \cdot \mathbf{x}_{21} \times \mathbf{x}_{31} \end{aligned} \quad (2)$$

Note that each cross product occurs twice, but obviously has to be computed only once. The times t_i when the four points are coplanar may now be computed as the the real roots of the polynomial

$$P(t) = a_3 \cdot t^3 + a_2 \cdot t^2 + a_1 \cdot t + a_0 \quad (3)$$

Since the coplanarity is a necessary but not sufficient condition for a collision, the distance between the primitives is computed for each time t_i, in ascending order. If the distance is below a certain threshold, a collision will be registered.

6.2 The Iterative Procedure

An iterative collision detection scheme suggested by Bridson et al. [2] works as follows: After the velocities of the particles for the next time step have been computed, the BVH is updated and traversed. When the bounding volumes of two leaf nodes overlap, the corresponding primitives are checked for close proximity. If their distance is below the material thickness, a collision response is applied.

Then the iterative procedure for the continuous collision detection starts. In each iteration, the BVH is updated so that the bounding volumes contain the start- and end position of the contained particles. Then the BVH is traversed, and for the primitives whose bounding volumes overlap, continuous collision tests are performed. All imminent collisions are detected, and collision responses are applied, which alter the candidate velocities of the particles. Then the next iteration starts. This step is repeated until no new collisions are detected, and the particles are free to move with the current velocities and without causing interpenetrations.

This process of iteratively traversing the whole BVH, registering the earliest collisions for all triangles, and updating the whole BVH may be very time-consuming. So we present a method which allows us to perform this task more efficiently.

A Single Iteration

After the iteration started, the BVH is traversed, and the continuous collision tests for the primitives of triangles with overlapping bounding volumes are performed, as described above. The distance between the primitives is computed for each time t_i, in ascending order. If the primitives actually collide at a time t_i, subsequent times $t_j > t_i$ are ignored. The collision for time t_i is registered, and associated with the particles it involves.

For each new collision test, the involved particles are examined. The earliest time t_0 of the collisions that already involve one of the particles is retrieved. Then, the times t_i for the new collision test are computed. But only for times $t_i < t_0$, the distance between the primitives has to be actually computed. When a collision at a time $t_i < t_0$ is registered, all collisions that are associated with particles of the collision that occurred at time t_0 are discarded. Thus, while in the method by Eberle [7] only the earliest collision for each *triangle* is treated in each iteration, we store the earliest collision for each *particle*, which still ensures that the collisions are treated in the correct chronological order.

Modification Marking

Usually, the number of collisions that are registered decreases rapidly within a few iterations. Thus, it is not necessary to update and traverse the whole BVH in each iteration, as possibly only very few particles received a collision response.

To prevent unnecessary updates and traversals, we suggest a scheme of "modification marking" for the BVH. First, we mark the particles that have been involved in a collision. Each particle stores the iteration in which it has been modified. This information is propagated bottom-up into the BVH, so that each node of the BVH contains the *last* iteration in which any of the particles it contains has been modified. In the next iteration, we update only the parts of the BVH that have been modified in the *previous* iteration. Additionally, the traversal may be restricted to those parts of the BVH in which new collisions may have occurred. So, during the traversal, we only check the bounding volumes of BVH nodes for overlap if at least *one* of the nodes contains a particle that has been modified in the previous iteration. Otherwise, the traversal may stop. This also allows us to skip the time-consuming collision tests that already failed in the previous iterations.

7 IMPLEMENTATION AND RESULTS

We have implemented the algorithms described in this paper in Java and employed them in a cloth simulation system. The patterns of the clothes are divided recursively, creating a BVH with degree 16. The bounding volumes for the BVH and for the primitives are 18-DOPs in each case. The tests were run on a standard Pentium 4 with 3.6 GHz and 3.25 GB RAM, using the Java Runtime Environment 1.5.0.

7.1 Test cases

The algorithms presented in this paper focus on accelerating the collision detection process for multiple layers of deformable triangle meshes in close proximity. Thus, we compared the speed of the different implementations with scenes involving multiple layers of cloth.

Cloth pile

Figure 3 shows images of 10 sheets of cloth falling on the floor, creating a pile of cloth with 10 layers. Each sheet consists of 100 particles and 164 triangles. The cloth is modeled with a realistic thickness of 1 mm. When all sheets lie on the floor, there are many primitives in close proximity. Note that the sheets are slightly shifted against each other at the beginning. Otherwise, the worst case depicted in Figure 2 would occur for *every* triangle, and the number of collision tests (and thus, the effect of the additional primitive bounding volumes) would be unrealistically high.

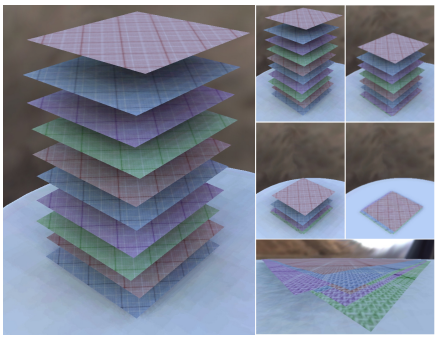

Figure 3: 'Cloth Pile': A pile of 10 sheets of cloth

Tumbling Torus

Figure 4 shows images of 50 sheets of cloth that fall upon a tumbling torus. The scene contains 5k particles and 8.2k triangles, and shows the interaction of high-velocity cloth with other layers: Many sheets are falling on the torus and hitting other layers, which are already moved and crumpled by the torus.

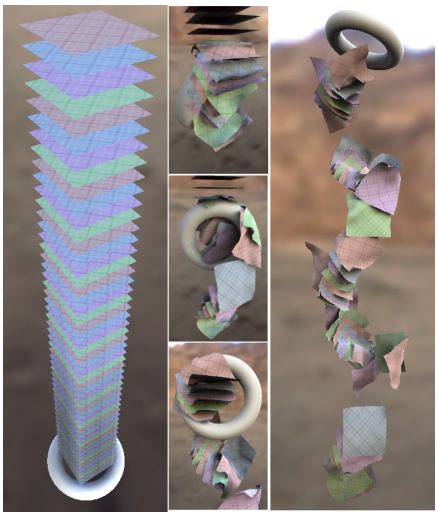

Figure 4: 'Tumbling Torus': A tumbling torus in a shower of 50 sheets of cloth

Garment

A scene from the garment simulation system described in the paper by Fuhrmann et al. [3] is shown in Figure 5. A woman is wearing a bodysuit, a blouse and trousers. The scene contains 8k particles and 13k triangles. In the area of the chest there are two layers of clothing, and three layers on the hips, where the trousers are pressing the other clothes against the body.

7.2 Results

We compared the speed of the overall collision detection process for the scenes described above. We also compared the number of collision tests to the number of collisions that actually occurred, and thus, the rate of 'false positives' that were reported by the BVH with

Figure 5: 'Garment': A woman wearing a bodysuit, a blouse and trousers

and without the additional bounding volumes for primitives.

For a usual cloth triangle mesh, our method on average requires two additional bounding volumes for each triangle, since #vertices + #edges $\approx 2 \cdot$ #triangles. But even for large triangle meshes, this memory overhead is small and justified by the speedup that can be achieved.

By default, we stored the information about the collision tests that had already been performed in the primitives, in order to avoid redundant collision tests. Compared to the total number of tests implied by overlapping triangle bounding volumes, this reduced the number of tests for all scenes by approximately one third, and thus complies with the results of the average case example from Figure 2.

Figure 6 shows the total time required for the collision detection in each frame in the 'Cloth Pile' scene. One can clearly see the times when each of the sheets falls on the pile of sheets that already lie on the floor, and that the time required for the collision detection increases with each layer.

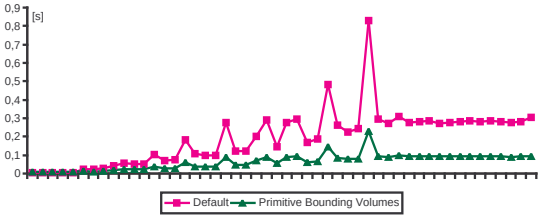

Figure 6: Total collision detection time per frame for scene 'Cloth Pile'

At the end of the simulation, when the sheets lie flat on each other, and no redundant collision tests are performed, approximately 5k collisions occur in each time step. Introducing additional bounding volumes for the primitives reduces the number of collision tests that are required to detect these collisions from more than 160k to only 22k. The ratio between the collision tests and the collisions that actually occur is shown in Figure 7.

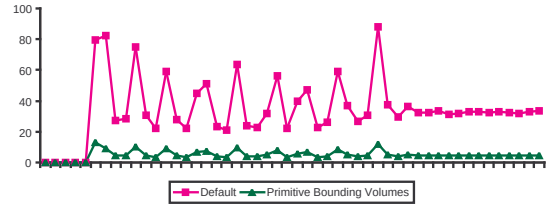

Figure 7: Ratio between number of collision tests and collisions for scene 'Cloth Pile'

Similar results can be achieved for the more complex scene, 'Tumbling Torus', as depicted in Figure 8. There are some situations where multiple layers of cloth hit other layers with high velocity, and a higher number of iterations is required to resolve all collisions, which causes the 'peaks' in the time required for the collision detection. Marking the parts of the BVH that contain particles which have been involved in a collision helps to avoid these peaks.

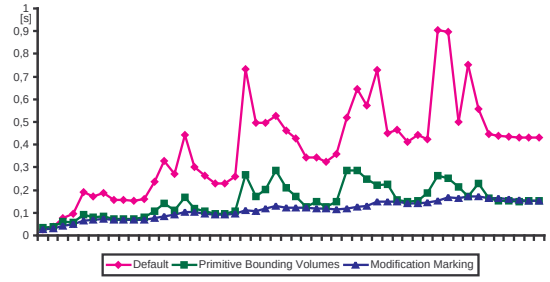

Figure 8: Total collision detection time per frame for scene 'Tumbling Torus'

The 'Garment' scene is the most complex and realistic test case. The three layers around the hips are pressed against the body, and many iterations are required to resolve all collisions. Restricting the update of the BVH and the collision tests to those parts that are actually modified by collision responses, by marking the modified parts of the BVH, brings a significant speedup in this scene, which can be seen in figure 9.

Performance

Table 2 summarizes the results for the three test cases. The additional bounding volumes cause a higher cost for updating the BVH. Additionally, more intersection tests between bounding volumes have to be performed. But the higher cost for updating the BVH is more than

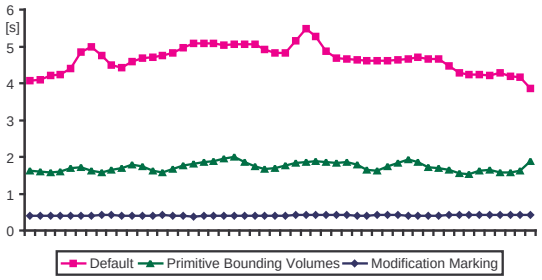

Figure 9: Total collision detection time per frame for scene 'Garment'

compensated by the benefit from the reduced number of exact collision tests, since an overlap test between bounding volumes is much cheaper than an exact collision test. Due to the reduced number of exact collision tests, the total time that is required for the collision detection in each frame can be reduced by a factor of 2.5 to 3.0.

The time that is required for the collision detection increases with the number of iterations that is necessary to resolve all collisions. By marking the modified parts of the BVH, the collision detection time less depends on the number of iterations, but more of the number of collisions that actually have to be resolved in each iteration. Thus, the complex garment scene runs more than 4 times faster, because only small parts of the cloth are involved in complex collision situations.

Scene	Default	PBV	Mod. Mark.
Cloth Pile	188	63	62
Tumbling Torus	379	150	110
Garment	4633	1713	407

Table 2: Average total time required for the collision detection in each frame, in milliseconds. PBV: With primitive bounding volumes. Mod. Mark.: With primitive bounding volumes and modification marking

8 CONCLUSION AND FUTURE WORK

We have shown how the process of continuous collision detection for deformable triangle meshes could be accelerated by the effective technique of introducing additional bounding volumes for the primitives of the triangle meshes. Making the broad phase a little bit more narrow saves a significant amount of computation time, especially for multiple layers of triangle meshes in close proximity. An efficient scheme for iteratively resolving collisions in complex collision situations yields plausible simulation results, and allows us to employ the robust, continuous collision detection even for complex scenes, with a frame rate suitable for interactive applications.

Future work will include the implementation of a more sophisticated collision response scheme. For ex-

ample, Volino and Magnenat-Thalmann [24] suggested a scheme which will probably help to decrease the number of iterations required for the collision detection. Examining the benefits of performing the updates and intersection tests of the primitive bounding volumes on the GPU could also be worthwhile, according to the research done by Greß et al. [6] on hierarchical collision detection. Another focus is on the creation of simulation states that are intersection-free, even when the initial state is not. This is crucial for applying the robust collision detection, as it would otherwise preserve and not remove the intersections. Further research also has to be done for a suitable user interaction, which does not violate the robustness, but still allows interactive draping of the simulated garments.

9 ACKNOWLEDGEMENTS

This work was part of the research grant KF0157401SS5 in the PRO INNO II program, partially funded by the German Federal Ministry of Economics and Technology (BMWi) via the Arbeitsgemeinschaft industrieller Forschungsvereinigungen "Otto von Guericke" e.V. (AiF). We thank our project partner Assyst GmbH for providing garment input data.

REFERENCES

[1] David Baraff, Andrew P. Witkin, and Michael Kass. Untangling cloth. *ACM Trans. Graph.*, 22(3):862–870, 2003.

[2] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. In *SIGGRAPH 02*, pages 594–603, 2002.

[3] Arnulph Fuhrmann, Clemens Gross, and Volker Luckas. Interactive animation of cloth including self collision detection. *Journal of WSCG*, 11(1):141–148, February 2003.

[4] Stefan Gottschalk, Ming C. Lin, and Dinesh Manocha. Obbtree: A hierarchical structure for rapid interference detection. In *SIGGRAPH*, pages 171–180, 1996.

[5] Naga K. Govindaraju, David Knott, Nitin Jain, Ilknur Kabul, Rasmus Tamstorf, Russell Gayle, Ming C. Lin, and Dinesh Manocha. Interactive collision detection between deformable models using chromatic decomposition. *ACM Trans. Graph.*, 24(3):991–999, 2005.

[6] A. Greß, M. Guthe, and R. Klein. Gpu-based collision detection for deformable parameterized surfaces. *Computer Graphics Forum*, 25(3):497–506, September 2006.

[7] Sunil Hadap, Dave Eberle, Pascal Volino, Ming C. Lin, Stephane Redon, and Christer Ericson. Collision detection and proximity queries. In *GRAPH '04: Proceedings of the conference on SIGGRAPH 2004 course notes*, New York, NY, USA, 2004. ACM Press.

[8] S. Kimmerle, Matthieu Nesme, and François Faure. Hierarchy accelerated stochastic collision detection. In *Vision, Modeling, and Visualization*, Stanford, California, 2004.

[9] James T. Klosowski, Martin Held, Joseph S. B. Mitchell, Henry Sowizral, and Karel Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Trans. Vis. Comput. Graph.*, 4(1):21–36, 1998.

[10] Thomas Larsson and Tomas Akenine-Möller. A dynamic bounding volume hierarchy for generalized collision detection. *Computers and Graphics*, 30(3):451–460, June 2006.

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

[12] Brian Mirtich and John F. Canny. Impulse-based dynamic simulation. Technical Report UCB/CSD-94-815, EECS Department, University of California, Berkeley, 1994.

[13] Ian J. Palmer and Richard L. Grimsdale. Collision detection for animation using sphere-trees. *Comput. Graph. Forum*, 14(2):105–116, 1995.

[14] Xavier 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.

[15] A. Sanna and M. Milani. Cdfast: an algorithm combining different bounding volume strategies for real time collision detection. In *SCI 2004 Proceedings*, volume 2, pages 144–149, 2004.

[16] Philip J. Schneider and David H. Eberly. *Geometric Tools for Computer Graphics*. Morgan Kaufmann, 2003.

[17] Avneesh Sud, Naga Govindaraju, Russell Gayle, Ilknur Kabul, and Dinesh Manocha. Fast proximity computation among deformable models using discrete Voronoi diagrams. *ACM Transactions on Graphics*, 25(3):1144–1153, July 2006.

[18] 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*, 24(1):61–81, March 2005.

[19] Matthias Teschner, Bruno Heidelberger, Matthias Müller, Danat Pomerantes, and Markus H. Gross. Optimized spatial hashing for collision detection of deformable objects. In *8th International Fall Workshop Vision, Modeling, and Visualization (VMV)*, pages 47–54, 2003.

[20] Gino van den Bergen. Efficient collision detection of complex deformable models using aabb trees. *journal of graphics tools*, 2(4):1–14, 1997.

[21] P. Volino and N. Magnenat-Thalmann. Developing simulation techniques for an interactive clothing system. In *Proceedings of International Conference on Virtual Systems and Multimedia*, pages 109–118. IEEE Computer Society, 1997.

[22] Pascal Volino, Martin Courchesne, and Nadia Magnenat-Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. In *SIGGRAPH 95*, pages 137–144, 1995.

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

[24] Pascal Volino and Nadia Magnenat-Thalmann. Accurate collision response on polygonal meshes. In *CA*, pages 154–163, 2000.

[25] Pascal Volino and Nadia Magnenat-Thalmann. Resolving surface collisions through intersection contour minimization. *ACM Transactions on Graphics*, 25(3):1154–1159, July 2006.

[26] Wingo Sai-Keung Wong and George Baciu. Dynamic interaction between deformable surfaces and nonsmooth objects. *IEEE Transactions on Visualization and Computer Graphics*, 11(3):329–340, 2005.

[27] Gabriel Zachmann and Jan Klein. Adb-trees: Controlling the error of time-critical collision detection. In *8th International Fall Workshop Vision, Modeling, and Visualization (VMV)*, pages 19–21, University München, Germany, November 2003.

[28] Gabriel Zachmann and René Weller. Kinetic bounding volume hierarchies for deformable objects. In *ACM Int'l Conf. on Virtual Reality Continuum and Its Applications (VRCIA)*, Hong Kong, China, June 14-17 2006.