Hierarchy Accelerated Stochastic Collision Detection

S. Kimmerle[1] M. Nesme[2] F. Faure[2]

[1]WSI/GRIS, Universität Tübingen, Germany
[2]GRAVIR, INRIA Rhône-Alpes, Grenoble, France
Email: kimmerle@gris.uni-tuebingen.de
{matthieu.nesme,francois.faure}@imag.fr

Abstract

In this paper we present a new framework for collision and self-collision detection for highly deformable objects such as cloth. It permits to efficiently trade off accuracy for speed by combining two different collision detection approaches. We use a newly developed stochastic method, where close features of the objects are found by tracking randomly selected pairs of geometric primitives, and a hierarchy of discrete oriented polytopes (DOPs). This bounding volume hierarchy (BVH) is used to narrow the regions where random pairs are generated, therefore fewer random samples are necessary. Additionally the cost in each time step for the BVH can be greatly reduced compared to pure BVH-approaches by using a lazy hierarchy update. For the example of a cloth simulation framework it is experimentally shown that it is not necessary to respond to all collisions to maintain a stable simulation. Hence, the tuning of the computation time devoted to collision detection is possible and yields faster simulations.

1 Introduction

Interactive simulation of complex deformable objects such as cloth, hair, or biological tissues still remains a challenge in computer graphics. Two main tasks are involved: computing the dynamics of the objects subject to internal and external forces, and detecting the collisions between the objects or within a single object. Though dynamics can now be efficiently handled by methods based on implicit integration [2], collision detection remains computational expensive due to the inherent quadratic complexity of the problem since each element may collide with any other one. Therefore acceleration structures for the collision detection are developed. Hierarchical space or object decomposition reduce the complexity to $O(n \log n)$, where n is the number of elements considered, typically triangles, edges, or vertices. Sorting elements along axes can lead to linear complexity in good cases, but remain quadratic in general. However, these approaches provide very little control of the computation time. Currently, when none of them is fast enough, interactive simulations are impossible.

Our goal is to provide a method which, given an arbitrary bound on runtime, handles the collisions as exact as possible, hence allowing interactive animation of arbitrarily large scenes. In this paper, we propose a new collision detection framework which allows us to easily trade-off accuracy for computation time. It is based on the combination of two approaches: the first manages a set of randomly chosen pairs of geometric primitives which iteratively converge to local distance minima; the second is based on a hierarchy of bounding volumes which helps us to pick appropriate random pairs.

The remainder of the paper is organized as follows. We first briefly summarize previous work. Section 3 presents the method based on the convergence of randomly chosen geometric primitives. The hierarchy of bounding volumes is presented in Section 4 before the combination of these two methods to the *Hierarchy Accelerated Stochastic Collision Detection* is explained in Section 5. Finally the obtained results and the validity of our approach is discussed.

2 Previous Work

Numerous collision detection methods for various purposes have been developed [18, 13]. Most of the work however focusses on convex polyhedra [17, 21], respectively methods for decomposing non-convex objects into convex parts [7]. However,

these approaches are not applicable for highly non-convex objects such as hair or cloth.

Currently, collision detection for non-convex objects is mostly accelerated by bounding volume hierarchies. The bounding volumes applied range from spheres [24, 22], oriented bounding boxes (OBBs) [10], axis aligned bounding boxes (AABBs) [6, 29], and other discrete oriented polytopes (DOPs) [14, 32] to shell trees [15].

Most of these approaches, however, are not designed for deformable models as they rely on precomputed data. Various researchers have tried to circumvent these problems and to adapt the methods to deformable objects by efficiently updating or rebuilding AABBs [29, 9]. It appears that AABBs perform better than more complex bounding volumes like OBBs when dealing with highly deformable objects since they can be faster updated. It has also been pointed out that higher order trees like 4-ary trees or 8-ary trees are more efficient than binary trees for deformable objects with many intersecting or close areas like in the case of cloth or human tissue, because fewer nodes need to be updated in each time step and the recursion depth during overlap tests is lower [27, 16, 20].

Besides bounding volume hierarchies recent collision detection methods for deformable objects have relied on distance fields [5, 8], graphics hardware or image-space techniques [1, 19, 30], and spatial subdivision methods, dividing the scene into voxels [33, 28].

Methods based on curvature and orientation to accelerate self-collision detection have also been presented for cloth simulation [31, 23]. Self-intersection needs special attention, since thin objects are subject to tunnelling effects due to time sampling. To avoid the tunnelling a continuous motion from the start to the end of each time step has to be considered as a whole rather than considering initial and final states separately. Continuous collision detection has been studied for both deformable [4] and rigid bodies [26].

Generally, a fast method is not enough for interactivity. Since realistic scenes can be arbitrarily large, all the methods mentioned above will eventually fail to provide interactive frame rates. To obtain interactivity, we need to allow the user to trade off precision for speed during the collision detection between deformable objects. A naive approach would simply focus on an arbitrarily reduced subset of the scene. Unfortunately, iteratively applied to a static scene, the method would repeat the same mistakes while a better method would eventually detect all existent collisions. Given the impossibility of examining all object pairs at each frame and the need to examine all pairs to detect all collisions, the exploitation of temporal coherence by reusing the previously obtained results is necessary.

Temporal coherence has long been applied to rigid bodies [6, 11]. Only recently methods employing temporal coherence for deformable objects have been presented [12, 25]. Moreover, they allow us to balance precision (i.e. completeness) and speed. We thus build upon such a method and try to circumvent its current drawbacks, which are later detailed. Our main contributions are: (1) an enhanced version of this approach, presented in the next section; (2) a new hybrid method to combine its advantages with the power of bounding volume hierarchies, presented in section 5.

3 Active Pairs

The goal of our new method called *Active Pairs* is to efficiently detect collisions between two meshes or self-collisions within a single mesh. An active pair may be any pair of geometric primitives (vertices, edges, triangles, etc.) of the concerned meshes. The presented method exploits temporal coherence between consecutive time steps and provides the user with the ability to trade detection completeness for computation speed. A method similar to our active pairs was originally developed in cooperation with Raghupathi et al. [25]. In the remainder of this section we therefore present the complete method and also detail our specific contributions.

During the collision detection process with the active pair method at first a set of active pairs is chosen randomly. For reasons of clarity, in the illustration of a self-collision of a deformable body (Figure 1) only point-point pairs are shown. Each of these pairs then iteratively converges to a local distance minimum by repeatedly replacing each element by its topological neighbor with smaller distance compared to the other elements (Figure 1(a)). If the distance falls below a given proximity threshold, a collision is detected. Often, collisions between deformable bodies are not isolated but they are grouped in collision clusters, each of them corresponding to a pair of colliding areas (Fig-

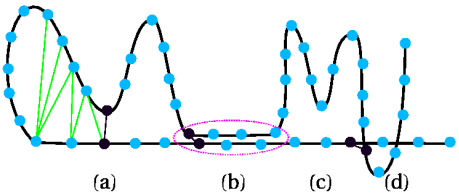

Figure 1: Self-collisions of a deformable object. The active pairs of points are shown in black. (a) An active pair has converged iteratively to a local distance minimum. (b) A collision has been detected and the associated collision cluster is encircled. (c) No active pair is present yet. (d) A collision has been detected too late and an intersection already occurred.

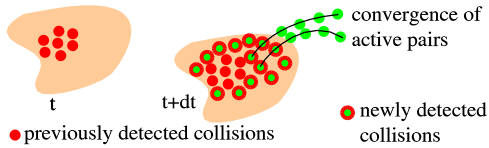

Figure 2: A collision cluster. Two new active pairs have converged (black lines) and thereby detected new collisions (around the lines). In the same time step the previously detected collisions have been updated and yield new detected collisions.

ure 1(b)). Especially for this case the active pair method rapidly finds all nearby collision pairs during the described convergence process. A local distance minimum where no active pair is present, such as in Figure 1(c), might eventually lead to an undetected collision (Figure 1(d)). The probability to create an active pair in a certain region before an intersection occurs depends in part on the number of new active pairs we create at each time step. If, during or at the end of a convergence process, the elements of an active pair are far from each other, or if they are moving apart, then we can consider that the probability of detecting a collision by this active pair in the near future is very low. This pair can thus be considered useless and discarded.

At each step of the animation loop, we update each active pair by iterating it until convergence is reached at a new local minimum resulting from the motion of the bodies. Due to temporal coherence in the animation, the number of necessary iterations is generally small. At each iteration of one pair, the neighboring pairs are visited for possible consecutive collisions and detected collisions are added to the list, as well as the possibly new closest pair replaces the current active pair. We actually store the visited pairs in a hash table and look for the closest not already visited neighboring pair to avoid tracking the same local minimum twice. Collision clusters are progressively detected either by growth around the currently detected collisions, or by the convergence of new pairs, as illustrated in Figure 2.

With the presented active pairs method, we can not guarantee to find all the collisions unless we try all possible pairs. Let us denote *detection ratio* the number of detected collisions divided by the number of actual collisions. This ratio as well as the computation time devoted to collision detection depends on the number of active pairs we use. By tuning the number of active pairs the trade-off between detection ratio (quality) and computation time (speed) is possible. In the remainder of this section we investigate how to obtain a good detection ratio and we discuss the consequences of incomplete collision detection.

Robustness is a major question, since we can not guarantee to detect all the collisions. In such a case the bodies can intersect each other (Figure 1(d)), and then it is important to ensure that the collision response does not suddenly react by a high amount of potential energy which would make the system unstable. When computing the reaction to a collision (edge-edge or vertex-triangle collision), we thus consider edges and triangles as inelastic rigid bodies with non-zero thickness and geometric constraints. The velocities are handled similarly to [4], while intersections are solved using the same method extended to positions. Due to object stiffness and damping, position and velocity corrections are then partly propagated to neighboring elements. An acceptable behavior is thus obtained in areas where a significant number of collisions are detected. Additionally, to handle intersections we could apply recent methods for untangling cloth [3].

The active pairs method described in this paper strongly improves previous stochastic approaches. While in [25], only pairs of edges are considered, we generalize to a variety of geometric primitives: combinations of points, edges and triangles. The only requirement is that the distance between two

primitives must be computable. In addition, voxels could also be used for volumetric objects. Additionally, we start the converging process by using pairs of points, since in this case distance computation is faster than for the other primitives. When the elements are close enough, we switch to the exact distances of the two primitives, since they are locally more accurate and provide more relevant information for collision handling.

In earlier work, for each new collision, the associated collision cluster was fully detected using a recursive search. This is extremely expensive and can be highly redundant, as several new colliding pairs may appear around the same cluster at each time step (Figure 2). As detailed in the results section, we found our presented approach experimentally much more efficient.

4 Bounding Volume Hierarchy

Using only a stochastic method to find new close regions between two meshes has a major drawback. The candidate pairs, where the iterative search for the close regions begins, are, by the design of the method, randomly distributed over the complete mesh. Nevertheless, often the close regions between two meshes are restricted to a rather small area of the complete mesh.

To overcome this drawback of the pure stochastic collisions detection we combined it with a bounding volume hierarchy of k-DOPs. In this section we detail the bounding volume hierarchy used, its advantages over other possible choices, and our strategies for accelerating the hierarchy update.

4.1 Bounding Volumes

We have chosen k-DOPs as bounding volumes, because they usually fit more tightly around meshes than spheres, while being much faster to update than OBBs [29].

A k-DOP [14] is a convex polyhedron defined by k halfspaces. The halfspaces are given by their normals which are the same for all k-DOPs. If the set of directions is chosen to be the positive and negative x-, y-, and z-axes, then k-DOPs are just AABBs.

The k-DOP bounding volume is built by inserting points into an initially 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 test, 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). In order to detect all k-DOP proximities within a distance ε, each k-DOP is enlarged by an offset of $\varepsilon/2$ in each of its k directions.

4.2 Hierarchy Generation

For k-DOP bounding volumes, as for AABBs, the optimal bounding volume for a node in the hierarchy can be easily computed by merging its child bounding volumes. Conversely the hierarchy can be efficiently built using a top-down splitting method. In our implementation first the longest side of a k-DOP is determined by the face pair with the maximum distance. Subsequently the k-DOP is split parallel to this face pair through its center. As some polygons are generally cut by the splitting plane, they are assigned to that child node which would contain the smaller number of polygons. In the lower hierarchy levels, if all polygons happen to be cut, each of them is assigned to its own node. Finally, as the corresponding vertices for the node are known, the k-DOPs can be optimally fitted to the underlying faces. Although this method is simple, it turns out to be efficient on the one hand and to produce well balanced trees on the other hand.

Previous approaches employed binary trees to store the hierarchy since they require the smallest number of overlap tests. However for deformable objects 4-ary trees or 8-ary trees have shown better performance [20]. This is mainly due to the fact that fewer nodes need to be updated and the total update costs are lower. Additionally, the recursion depth during overlap tests is lower and therefore the memory requirements on the stack are lower. For our results we have chosen 4-ary trees and did not see significant differences to 8-ary trees.

Generally, the hierarchy update re-inserts the vertices into the leaf k-DOPs and builds the inner k-DOPs by unifying the $k/2$ intervals of the child bounding volumes. Because this hierarchy update can be costly we use a lazy hierarchy update. In this scheme, parts of the hierarchy, where vertices did not move more than a certain distance, can be omitted during the hierarchy update. Thus, the hierarchy update is accelerated for slow parts of the scene and for small time step sizes. As detailed in Section 5

this lazy update perfectly fits to our stochastic approach.

5 Combined Method

Having presented the separate methods in the previous sections, we now will focus on the combination of stochastic collision detection and a k-DOP hierarchy.

Bounding volume hierarchies have been shown to be very efficient for collision detection of deformable objects [29, 16, 20]. Nevertheless they have two major drawbacks. First, the hierarchy needs to be updated in every time step due to the deformations of the objects, which is very time consuming. Second, bounding volume hierarchies are usually built around triangles or tetrahedrons, and return colliding or nearby pairs of these. Nevertheless an efficient and physically correct collision response needs to be employed on the vertex level. Therefore, in a second collision detection step, all close triangle pairs are checked for proximities between two edges, or a vertex and a triangle, where the necessary collision response can be easily applied.

In the previously presented stochastic collision detection, the collision detection is carried out directly on the primitives; to do so, edge-edge and vertex-triangle distances have to be computed. For most collision response schemes this measurements are needed as well. However the drawback of the stochastic collision detection is that there exists no control at all where the random pairs are placed. Due to this equal distribution across the whole collision objects, most random sample pairs may be generated in areas without any collision.

Our new approach combines this stochastic collision detection with a bounding volume hierarchy. In this combination, the drawbacks of both methods can be minimized. The process involves two steps: first a collision detection on the k-DOP -hierarchy is performed, returning all colliding leaves of the hierarchy tree. Each leaf can contain a specified number of triangles or other primitives. Then in the second step, a stochastic collision detection as described in Section 3 is performed, where the random sample pairs are created only on the colliding leaves.

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

update(A and B) buttom-up as detailed in Section 4.2
traverse(A,B)
if A and B do not overlap **then**
  return
end if
if A and B are leaves **then**
  return intersecting leaves (DOPs)
else
  for all children A[i] and B[j] **do**
    traverse(A[i],B[j])
  end for
end if
Stochastic Detection as detailed in Section 3, while picking the active pairs out of the intersecting leaves of A and B.

6 Results

To show that a detection ratio below 100% is sufficient to ensure a stable simulation we used the bounding volume hierarchy method (100% detection ratio) and carried out the collision response only for randomly selected collisions. Even for a very challenging cloth simulation (very thin object, i.e., high risk of interpenetrations) we experimentally found that a response (detection) ratio as low as 20% can be enough (Figure 4) to get good visual results.

6.1 Comparison with other methods

We compared our method with two other methods: a simple stochastic approach as described in Section 3 without hierarchy acceleration and an efficient hierarchical bounding box method based on k-DOPs as described in Section 4. As a test framework, we incorporated the collision detection module into a cloth simulation system.

For a static scene (torus and ellipsoid) we show that the detection ratio of our presented method rapidly converges to 100% (Figure 3). The convergence is much faster than with a pure stochastic method, because all randomly picked pairs are already close to local minima. To achieve the same speed of convergence with the pure stochastic method, many more random pairs are necessary.

For a dressed woman (Figure 4) and a dressed man (Figure 5) the combined hierarchy accelerated stochastic collision detection shows a much lower collision detection time than the pure bounding volume hierarchy or the stochastic method (see Table 1 and 2). This is because it is not necessary to recompute all collisions in every time step; instead, one

obtains temporal coherence by keeping a list of collisions from step to step. Beyond that, the stochastic method is directly employed on the colliding primitives (vertices, edges, triangles, etc.) and no further computation is needed as for the pure hierarchical method. Additionally, as detailed in Section 5, the hierarchy update is faster and the detection of colliding DOPs can be faster as more faces are put into a DOP and therefore fewer DOPs are in the hierarchy. Compared to the pure stochastic method the combined framework achieves a much higher detection ratio with the same number of random samples, because they are only generated in "definitely close" areas. We note that in our set-up the pure stochastic method needs about three times as much random samples to achieve the same detection ratio.

We also challenged our presented method with a complex catwalk animation. Therefore we computed this dynamic scene with two different setting of our method. Once with 100 new active pairs per time step and once with 50 new active pairs per time step. During the animation we thereby achieved collision ratios between 40% and 80% (Figure 6(b)) respectively 20% and 70% (Figure 6(c)). For both settings the catwalk simulation remains stable during the complete time of 16 seconds (400 frames or 1600 time steps). Single frames of this simulation are compared to the results of a collision detection with a pure BVH (Figure 6(a)) and a detection ratio of 100%. The corresponding calculation times are compared in Table 3 and show that without a significant loss of quality a speed-up of a factor two is possible. With less quality an even larger speed-up was achieved.

All measurements of computation times in this paper were made on a 2.4 GHz Pentium IV with 1GB RAM.

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

Table 1: Collision detection times for the dressed woman (avatar 53000 faces, dress 3800 faces) measured in seconds for the simulation of 100 time steps.

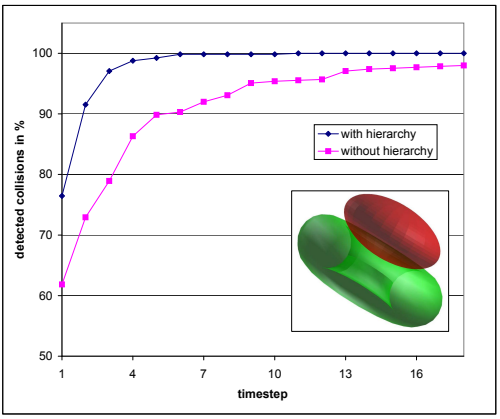

Figure 3: Detection ratio depending on time step. For same number of randomly created candidate pairs per time step, the presented method converges much faster than the pure stochastic method.

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

Table 2: Collision detection times for the dressed man (avatar 25000 faces, shirt 4400 faces, trousers 4000 faces) measured in seconds for the simulation of 50 time steps.

6.2 Discussion and future work

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

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

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

Table 3: Collision detection times for the catwalk animation (avatar 53000 faces, dress 3800 faces) measured in seconds for the simulation of 1600 time steps.

a certain collision ratio for a specified number of active pairs (as with any other non-exact method).

In the future, we plan to apply this framework to other kinds of geometric primitives, such as voxels, to implement continuous detection to avoid tunnelling effects, model and handle the collision clusters more efficiently, strictly bind the computation time, and design reaction methods more suited to scattered collision modelling and late intersection discovery. Additionally it would be useful to find a more theoretical approach to these new techniques to be able to dynamically set the number of random pairs or to predict the collision ratio.

6.3 Conclusion

We have presented a new framework for collision detection between deformable objects. It is the first to efficiently allow a tunable trade-off between accuracy and computation time. Since collision detection is the main bottleneck in the animation of deformable surfaces, we believe that it is a first step toward an efficient control of computation time in interactive applications.

Acknowledgements

For the cloth models we would like to thank the project partners of the bmb+f project "Virtual Try-On".

This work was supported by a grant from the Ministry of Science, Research and the Arts of Baden-Württemberg (AZ23-7532.24-34-14/2) and by a grant from the German Academic Exchange Service to Stefan Kimmerle.

References

[1] 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.

[2] D. Baraff and A. Witkin. Large Steps in Cloth Simulation. In *Proceedings of ACM SIGGRAPH*, pages 43–54, 1998.

[3] D. Baraff, A. Witkin, and M. Kass. Untangling cloth. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003)*, 22(3):862–870, 2003.

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

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

[6] 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.

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

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

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

[10] S. Gottschalk, M. Lin, and D. Manocha. OBBTree: A hierarchical structure for rapid interference detection. In H. Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, pages 171–180. ACM SIGGRAPH, Addison Wesley, Aug. 1996.

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

273, 1999.

[12] S. Guy and G. Debunne. Layered shells for fast collision detection, 2002. Unpublished.

[13] P. Jimenez, F. Thomas, and C. Torras. 3d collision detection: A survey. *Computers & Graphics*, 25(2):269–285, 2001.

[14] J. Klosowski, M. Held, J. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. In *Proc. of SIGGRAPH '96*, pages 171–180, 1996.

[15] S. Krishnan, A. Pattekar, M. C. Lin, and D. Manocha. Spherical Shells: A Higher-Order Bounding Volume for Fast Proximity Queries. Technical report, Department of Computer Science, University of North Carolina, 1997.

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

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

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

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

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

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

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

[23] 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.

[24] S. Quinlan. Efficient distance computation between non-convex objects. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 3324–3329, 1994.

[25] L. Raghupathi, V. Cantin, F. Faure, and M.-P. Cani. Real-time Simulation of Self-collisions for Virtual Intestinal Surgery. In N. Ayache and H. Delingette, editors, *Surg. Sim. & Soft Tis. Model.*, LNCS no. 2673, pages 15–26. Springer, Heidelberg, 2003.

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

[27] A. Smith, Y. Kitamura, H. Takemura, and F. Kishino. A simple and efficient method for accurate collision detection among deformable polyhedra. In *Proceedings of IEEE Virtual Reality Annual International Symposium*, pages 136–145, 1995.

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

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

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

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

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

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

Figure 4: Dressed woman (avatar 53000 faces, dress 3800 faces) with 100%, 20% and 10% collision response ratio.

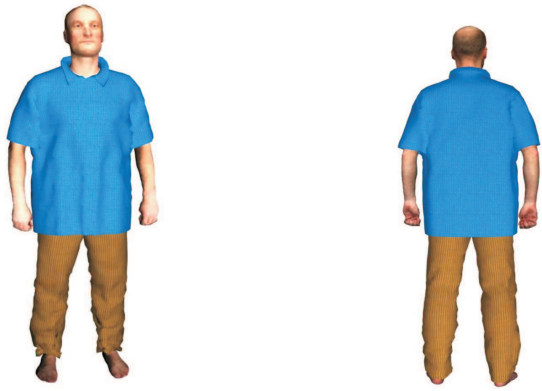

Figure 5: Example of stochastic collision detection and response in a cloth simulation (avatar 25000 faces, shirt 4400 faces, trousers 4000 faces).

(a) (b) (c)

Figure 6: Catwalk animation calculated with a BVH collision detection (a), and the presented Hierarchy Accelerated Stochastic Collision Detection (b) and (c) (avatar 53000 faces, dress 3800 faces).