The Visual Computer manuscript No.
(will be inserted by the editor)

Min Tang · Sung-Eui Yoon · Dinesh Manocha

Adjacency-based Culling for Continuous Collision Detection

Abstract We present an efficient approach to reduce the number of elementary tests for continuous collision detection between rigid and deformable models. Our algorithm exploits the connectivity information and uses the adjacency relationships between the triangles to perform hierarchical culling. This can be combined with table-based lookups to eliminate duplicate elementary tests. In practice, our approach can reduce the number of elementary tests by two orders of magnitude. We demonstrate the performance of our algorithm on various challenging rigid body and deformable simulations.

Keywords Adjacency based culling · Continuous collision detection · Elementary test · Duplication elimination

1 Introduction

Continuous collision detection (CCD) is frequently used for dynamic simulation of rigid and deformable models [20]. Given two discrete positions of an object or a primitive, a CCD algorithm computes an interpolating continuous trajectory between those instances (e.g. linear interpolation) and checks for collisions of the resulting swept volumes with other primitives. The main goal is to ensure that there are no collisions between the two discrete instances. As compared to discrete collision detection, CCD is much more expensive [2,13]. Specifically, the problem of performing CCD computation between two triangles undergoing linearly interpolated motion reduces to performing 15 elementary tests between edge/edge or vertex/face features of the two triangles. Each of these elementary tests reduces to solving the roots of a cubic equation.

Most collision detection algorithm use bounding volume hierarchies (BVHs) to reduce the number of CCD tests between triangle pairs [2,5,9]. However, BVHs are unable to cull away a high number of pairs and thereby result in a high number of false positives. As a result, current CCD algorithms spend a significant fraction of the collision query time in performing exact elementary tests between the features. There is considerable recent work on reducing the number of pairwise feature tests that either use bounding volumes for features or utilize connectivity information or generate separate hierarchies for these features [3,8,6].

Main contributions: In this paper, we address the problem of reducing the number of pairwise feature tests by exploiting the connectivity of the mesh. These include elementary feature tests between adjacent triangles that either share an edge or a vertex and are not culled by the BVHs. The other issue is duplicate tests that arise as these features are shared among multiple potentially colliding triangle pairs (PCTPs).

We present a hierarchical triangle-based culling method that exploits the adjacency information of PCTPs. Moreover, this formulation is combined with table based duplication elimination scheme to significantly reduce the number of elementary tests. Our approach can be combined with any BVH and is relatively simple to implement. We have tested its performance on different benchmarks corresponding to deformable models and multi-body simulations. In practice, we observe a reduction in the number of elementary tests by two orders of magnitude and an improvement in the overall performance by one order of magnitude.

Organization: The rest of the paper is organized as follows: Sec. 2 gives a brief survey of prior work. We introduce our notation and describe the overall pipeline of our approach in Sec. 3. The adjacency based culling

M. Tang
Zhejiang University, China
University of North Carolina at Chapel Hill, USA
E-mail: tangm@cs.unc.edu

S. Yoon
Korea Advanced Institute of Science and Technology (KAIST), South Korea
E-mail: sungeui@cs.kaist.ac.kr

D. Manocha
University of North Carolina at Chapel Hill, USA
E-mail: dm@cs.unc.edu

technique is described in Sec. 4. Sec. 5 presents the table based duplication elimination scheme. We describe our implementation and highlight its performance on various benchmarks in Sec. 6. We compare our approach with other algorithms in Sec. 7.

2 Related Work

In this section, we give a brief overview of prior work on continuous collision detection and self-collision detection for deformable objects.

2.1 Continuous Collision Detection (CCD)

CCD algorithms are used to compute the first time of contact during the time interval defined by discrete time steps. They are frequently used for dynamic simulation [1,26] and robot motion planning [4,14,12]. But due to their high computation complexity, most prior interactive CCD algorithms are limited to rigid models [16] or articulated models [17,26]. Moreover, most local motion planning algorithms only perform discrete collision checking along a continuous path [25]. Many efficient algorithms for CCD between deformable models have also been proposed based on GPU-based computations [6,18,7] or bounding volume hierarchies [20,3,8]. At a high level, various CCD can be classified as the following types:

- **Triangle-based CCD:** The deformable objects are decomposed into triangles, and CCD is performed by checking all the PCTPs. When PCTPs are computed, elementary tests associated with all these triangle pairs are performed to find out the first time of contact between the features [6,8,22].
- **Feature-based CCD:** The deformable objects are treated as sets of features (vertices, edges, and faces). The collisions are computed by directly performing elementary tests among all these features [23,3]. The randomized marking scheme, [23] ensures that all the elementary tests between features will be performed only once. Therefore, duplications are avoided. [3] extends this idea by using more compact encoding scheme. It also uses "Representative Triangles" to build BVHs based on features instead of on triangles. With the help of bounding volumes of features, more false positives are culled.

Our approach is compared with above techniques in Sec. 7.

2.2 Self-Collision Detection

Comparing to rigid models or articulated models, the efficiency of CCD for deformable models is mainly governed by the cost of performing self-collisions. Due to the random nature of deformation, self-collisions need to be checked during each time step of the simulation. The self-collisions can be further classified as two types: self-collision between adjacent triangles, and self-collision between non-adjacent triangles [6]. The detection of the second type of collisions can be accelerated by using standard BVH techniques, In some cases, self-collisions between adjacent triangles are ignored. However, in many cases such as cloth simulation, missing these collisions can result in noticeable artifacts in the simulation. One such example is shown in Fig. 1, where two triangles share an edge and result in a collision along non-adjacent features.

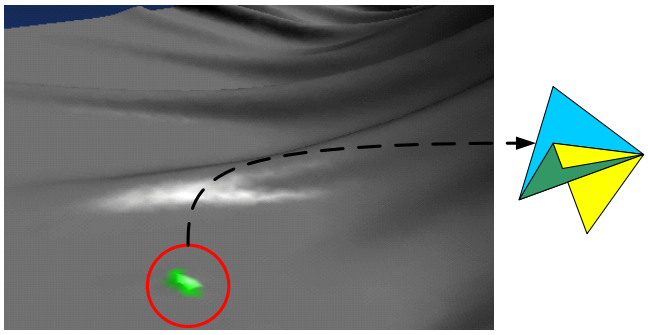

Fig. 1 Self-collision between two adjacent triangles that share an edge. Ignoring such collisions can effect the accuracy of cloth simulation

For discrete collision detection problem, [21] uses curvature criteria to remove self-collision free areas, and [15,22] makes further improvements to that formulation. Recently, [19] extend these ideas to continuous collision detection and presents a continuous normal cone (CNC) technique. In this formulation, all the normal cones are updated in a bottom-up manner during each simulation time step, and continuous contour tests are applied at the nodes of BVH to perform self-collision tests.

3 Overview

In this section, we introduce our notation and give an overview of our approach.

3.1 Notations

We use the symbols V, E, F, and T to represent vertices, edges, faces, and triangles, respectively. Lower-case symbols, v, e, f, and t are used to denote a specific vertex, edge, face, and triangle, respectively. Also, $\{t_a, t_b\}$ stands for a triangle pair of two triangles: t_a and t_b , and it is order independent, e.g., $\{t_a, t_b\} = \{t_b, t_a\}$.

Table 1 shows some statistic data about the ratios of adjacent triangle pairs and non-adjacent triangle pairs with overlapping bounding volumes in the benchmarks

Table 1 Ratios of adjacent and non-adjacent triangle pairs with overlapping bounding volumes (k-DOPs)

Benchmarks	Adjacent triangle pairs	Non-adjacent triangle pairs
Cloth-ball (Fig. 7)	88%	12%
N-body (Fig. 8)	84%	16%
Letters (Fig. 10)	93%	7%
Dancer (Fig. 9)	93%	7%

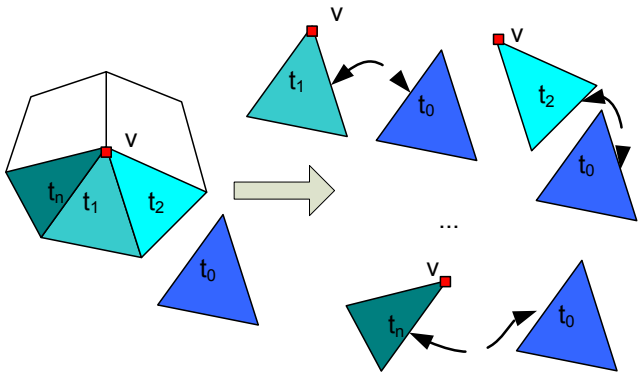

Fig. 2 Duplications in elementary tests: the Vertex/Face test $\{v, t_0\}$ is tested for n times when (t_1, t_0), (t_2, t_0), $\ldots$, (t_n, t_0) are processed respectively.

we used. As shown by the ratios, processing adjacent triangle pairs takes a major portion during the computation of CCD.

Table 1 shows some statistic data about the ratios of adjacent triangle pairs and non-adjacent triangle pairs with overlapping bounding volumes in the benchmarks we used. As shown by the ratios, adjacent triangle pairs play predominant roles in the computation of CCD.

Consider an example corresponding to the vertex/face test $\{v, t_0\}$ as an example (Fig. 2). This vertex/face test is tested repeatedly for n times when the PCTPs (t_1, t_0), (t_2, t_0), $\ldots$, (t_n, t_0) are processed respectively.

We classify all the PCTPs into three categories according to the adjacency of its two triangles:

- **Adjacent vertex triangle pair (AVTP):** This pair refers to two triangles that share one and only one vertex.
- **Adjacent edge triangle pair (AETP):** This pair refers to two triangles that share an edge.
- **Non-adjacent triangle pair (NTP):** This pair refers to two triangles that not share any vertex or edge.

All the feature pairs need to be checked for collisions during each simulation time step. We classify the colliding feature pairs into the following two types.

- **Static potential colliding feature pairs:** These feature pairs are generated as part of a preprocess from all the adjacent triangle pairs (AVTPs & AETPs). They are gathered once and remain unchanged during the whole process of simulation. After gathering all these feature pairs, the adjacent triangle pairs are ignored during subsequent simulation time steps. The gathering of static potential colliding feature pairs from the adjacent triangle pairs is referred to as *adjacency based culling*, and it will be explained in detail at Sect. 4.
- **Dynamic potential colliding feature pairs:** These feature pairs are gathered from those NTPs whose bounding volumes overlap, and are updated dynamically during each simulation time step.

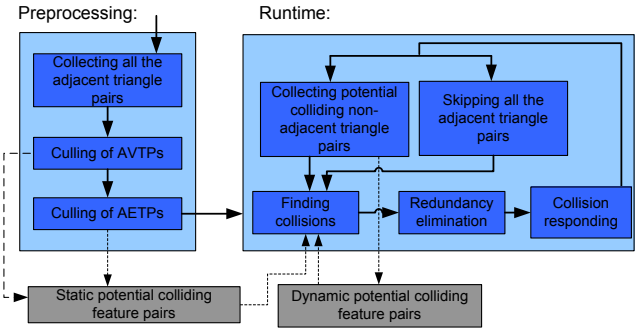

Fig. 3 Overall pipeline of our algorithm (solid arrows stands for control flow, and dashed arrows for data flow).

These feature pairs are gathered from those NTPs which pass bounding volume tests, and are updated dynamically at each simulation time step.

3.2 Overall Pipeline

As shown in Fig. 3, our algorithm consists of two stages: a preprocessing stage and a runtime stage.

During the preprocessing stage, all the adjacent triangles pairs are collected, and a hierarchical culling method is used to select potential colliding feature pairs (those passed boundary box tests) associated with adjacent triangle pairs. The culling results are stored as a set of static potential colliding feature pairs. In practice, these feature pairs are a very small fraction of the number of feature pairs associated with adjacent triangle pairs (0.2% for the "Cloth-ball" benchmark in Fig. 7, and 0.15% for the "Dancer" benchmark in Fig. 9).

At the runtime stage, all the adjacent triangle pairs are initially ignored in terms of pairwise feature tests. Only non-adjacent triangle pairs need to be checked for collisions. For non-adjacent triangle pairs whose bounding volumes overlap, the corresponding feature pairs (VF or EE) are collected and stored as a set of dynamic potential colliding feature pairs.

During each simulation time step, static potential colliding feature pairs and dynamic potential colliding feature pairs are checked for collisions based on exact elementary tests. We also use a table based duplication elimination method to remove all the duplications in the elementary tests (as explained in Sec. 5).

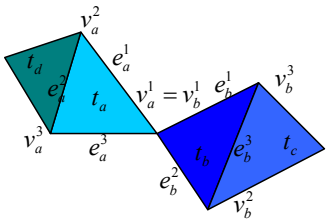

Fig. 4 Features related to an AVTP $\{t_a, t_b\}$.

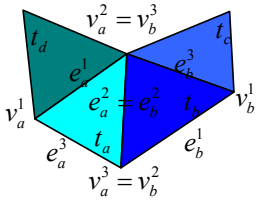

Fig. 5 Features related to an AETP $\{t_a, t_b\}$.

4 Adjacency-based Culling

As shown in Table 1, a significant fraction of overall collision query is spent on adjacent triangle pairs. In this section, we present a hierarchical triangle-based culling method named adjacency-based culling, which is capable of significantly reducing the number of elementary tests.

We process all the PCTPs hierarchically using the following three phases: NTP testing phase, AVTP testing phase, and AETP testing phase. At each processing stage, the elementary tests that been performed by previous processing stage(s) are skipped, i.e., AVTP related elementary tests are culled by processing of NTPs, and AETP related elementary tests are culled by the processing of NTPs and AVTPs. The culling of AVTP and AETP feature pairs in explained in detail in the subsequent sections.

4.1 Culling AVTP-related Feature Pairs

For an AVTP, 9 elementary tests (5 Edge/Edge and 4 Vertex/Face) need to be performed [6]. Since all the NTPs are always tested during each simulation time step, all the elementary tests that were performed during the NTP testing phase can be skipped. The culling rule is formulated based on Theorem 1.

Theorem 1 *Given four triangles, t_a, t_b, t_c and t_d, as shown in Fig. 4, t_a and t_b share one and only one vertex, $v_a^1 = v_b^1$, and form an AVTP $\{t_a, t_b\}$. Triangle t_d shares an edge with t_a, but does not share any vertex with t_b. Triangle t_c is defined symmetrically. If t_c and t_d exist, t_b does not share any vertex with t_d, and t_c does not share any vertex with t_a, then all the 9 elementary tests of AVTP $\{t_a, t_b\}$ can be skipped.*

Proof Let CCD(t_a, t_b) indicate all the 15 elementary tests need to be performed for an AVTP $\{t_a, t_b\}$. It consists of 6 $EE()$ tests and 9 $VF()$ tests. Let CCD_{sub} stand for a sub-set of all these 15 tests. Then CCD(t_a, t_b) can be decomposed into two sub-sets, $CCD_{sub}(t_d, t_b)$ and $CCD_{sub}(t_a, t_c)$:

$$\begin{aligned} CCD(t_a, t_b) &= CCD_{sub}(t_d, t_b) + CCD_{sub}(t_a, t_c) \\ CCD_{sub}(t_d, t_b) &= VF(v_a^3, t_b) + VF(v_a^2, t_b) + \\ &\quad EE(e_a^2, e_b^1) + EE(e_a^2, e_b^2) \end{aligned}$$

$$\begin{aligned} CCD_{sub}(t_a, t_c) = VF(v_b^3, t_a) + VF(v_b^2, t_a) + \\ EE(e_a^1, e_b^3) + EE(e_a^3, e_b^3) + EE(e_a^2, e_b^3). \end{aligned}$$

Since t_c and t_d exist, t_b does not share any vertex with t_d, and t_c does not share any vertex with t_a, there must exist two NTPs $\{t_a, t_c\}$ and $\{t_d, t_b\}$. Because all the elementary tests of $CCD_{sub}(t_d, t_b)$ and $CCD_{sub}(t_a, t_c)$ are already covered at NTP test phase, $CCD(t_a, t_b)$ can be skipped. □

If this theorem is satisfied, all the 9 elementary tests can be culled away. Otherwise, depending on the case where this theorem fails, the algorithm will record the feature pairs corresponding to $CCD_{sub}(t_d, t_b)$, $CCD_{sub}(t_a, t_c)$, or both of them as static potential colliding feature pairs.

This theorem can be interpreted in a geometric manner. It implies that for situations that the AVTP is on the boundary (t_c or t_d does not exist) or there are multiple adjacencies (t_b shares at least a vertex with t_d, or t_c shares at least a vertex with t_a), almost all the AVTP related feature pairs are already covered by NTP test phase. So in most cases, these feature pairs can be skipped. For example, in the "Cloth-ball" benchmark (Fig. 7), we are able to cut down 99.8% of AVTP related feature pairs, and in the "N-body" benchmark (Fig. 8), all the AVTP related feature pairs are culled based on this theorem.

4.2 Culling of AETP-related Feature Pairs

For an AETP, 4 elementary tests (2 Edge/Edge and 2 Vertex/Face) need to be performed [6]. Similar to the culling of AVTP related feature pairs, we can use the following theorem to cull away elementary tests that have been already covered by NTP testing phase and AVTP testing phase.

Theorem 2 *Given four triangles, t_a, t_b, t_c and t_d, as shown in Fig. 5, t_a and t_b share an edge, $e_a^2 = e_b^2$, and form an AETP $\{t_a, t_b\}$. Triangle t_d shares an edge with t_a, and shares an vertex with t_b. Triangle t_c is defined symmetrically. If t_c and t_d exist, then all the 4 elementary tests of AETP $\{t_a, t_b\}$ can be skipped.*

Proof Let $CCD(t_a, t_b)$ denotes the elementary tests needed to be performed for an AETP $\{t_a, t_b\}$. Then, $CCD(t_a, t_b)$

can be represented as follows:

$$CCD(t_a, t_b) = CCD_{sub}(t_a, t_c) + CCD_{sub}(t_d, t_b)$$
$$CCD_{sub}(t_a, t_c) = VF(v_b^1, t_a) + EE(e_a^3, e_b^3)$$
$$CCD_{sub}(t_d, t_b) = VF(v_a^1, t_b) + EE(e_a^1, e_b^1).$$

According to the definition of t_c and t_d, if t_c and t_d exist, they form two AVTPs: $\{t_a, t_c\}$ and $\{t_d, t_b\}$. Because all the elementary tests of $CCD_{sub}(t_a, t_c)$ and $CCD_{sub}(t_d, t_b)$ are already covered at AVTP test phase, $CCD(t_a, t_b)$ can be skipped. □

If this theorem is satisfied, all the 4 elementary tests in AETP can be skipped. Otherwise, depending on the reason of the failure of this theorem, we will record the feature pairs corresponding to $CCD_{sub}(t_a, t_c)$, $CCD_{sub}(t_d, t_b)$, or both of them as static potential colliding feature pairs.

The geometric meaning implied by above theorem is that, except for the situations where the AETP is on the boundary (t_c or t_d does not exist), almost all the AETP related feature pairs are already covered by AVTP test phase. So for most cases, these feature pairs can be skipped. For example, in the"Dancer" benchmark (Fig. 9), we can cull away more than 99.82% of AETP related feature pairs, and in the "N-body" benchmark (Fig. 8), almost all the AETP related feature pairs are culled away due to the theorem.

4.3 Handling Topological Changes

The formulation described above assumes that the topology of the mesh is fixed. In these cases, the gathering of static potential colliding feature pairs is performed only once during the preprocessing stage. Also, the static potential colliding feature pairs remains unchanged during all the runtime steps.

For deformable objects with dynamic topology, e.g., breaking or merging triangle meshes, we need to update the static potential colliding feature pairs during those time steps. In some cases, this update can be performed in an incremental manner.

5 Table-based Duplication Elimination

By using adjacency-based culling, the number of static potential colliding feature pairs is greatly reduced. However, the overall algorithm still lands up performing a high number of duplicate elementary tests among feature pairs, including static as well as dynamic potential colliding feature pair sets.

In order to eliminate duplicate tests, we use a table-based duplication elimination method. A feature test table is maintained by storing feature pairs as $[\{e_i, e_j\}, r_{ij}]$ or $[\{v_k, t_l\}, r_{kl}]$, where r_{ij} and r_{kl} are the elementary test results of the feature pairs $\{e_i, e_j\}$ and $\{v_k, t_l\}$ respectively.

For a feature pair that needs to be tested for exact collision test, we first search the pair in the feature test table. If the feature pair has been tested, the stored result is returned. Otherwise the cubic equation solver is invoked to compute the time of contact between the feature pair. Then the time of contact is saved into the feature test table and returned as a result.

The table-search strategy is quite simple yet effective. By assigning each feature (i.e., edge, vertex, and triangle) a unique id, the feature test table can be implemented as a hash table. In our current benchmarks, the hash table implementation is quite efficient in terms of removing all the duplicates. Since a large portion of false positives has been cut down by using adjacency-based culling, the table-based duplication elimination reduces many other feature pairs. For the "Cloth-ball" benchmark simulation in Fig. 7, 73.6% of the elementary tests are duplicated, and for the "Letters" benchmark in Fig. 10, 78.1% of the elementary tests are removed as duplications.

6 Implementation and Results

We have implemented our algorithm on a Windows/Vista platform using C++. All the timings are collected on a 2.66 GHz Intel Pentium machine with 2GB RAM, using a single thread. The elementary tests are evaluated by solving cubic equations numerically. In practice, it takes about 0.2 microseconds on average for each elementary test.

6.1 Dynamic BVH for Deformable Objects

A 2-level BVH is used as the acceleration structure for a scene consisting of deformable objects. The 2-level BVH is made up of an upper BVH and a set of lower BVHs. Specifically, all the deformable objects are decomposed into triangle meshes, each with a constant topological structure. We first compute a BVH for these triangle meshes, and then construct the upper BVH based on the bounding volumes. For each triangle mesh, which retains constant topological structure, a lower BVH is built with triangles at leaf nodes. More details about the 2-level BVH is described in [19]. In practice, the 2-level BVH gets similar or even better performance comparing to other BVH representations for deformable objects [2, 10, 11, 24].

6.2 Benchmarks

We used different benchmarks to test the performance of our algorithm, including two cloth simulation related scenes: "Cloth-ball" (Fig. 7) and "Dancer" (Fig. 9), and two N-body simulation related scenes: "N-body" (Fig. 8) and "Letters" (Fig. 10). At each simulation time step,

Table 2 Model complexity and average CCD time per frame

Benchmarks	#Tris	#Vertices	Average CCD Time per frame
Cloth-ball	$92K$	$47K$	$246ms$
Dancer	$40K$	$20K$	$37ms$
N-body	$34K$	$18K$	$82ms$
Letters	$5K$	$2K$	$9ms$

Table 3 Number of elementary tests

Benchmarks	W/O adjacency-based culling		With adjacency-based culling	
	AVTP	AETP	AVTP	AETP
Cloth-ball	$3.83M$	$551K$	7790	986
Dancer	$1.88M$	$239K$	2705	442
N-body	$1.31M$	$205K$	0	10
Letters	$196K$	$28K$	171	49

Table 4 Efficiency of duplication elimination method

Benchmarks	Without duplication elimination	With duplication elimination	Ratio
Cloth-ball	$173M$	$43M$	26.4%
Dancer	$15M$	$4.1M$	26.4%
N-body	$65M$	$22.6M$	34.6%
Letters	$20M$	$4.4M$	21.9%

we use our CCD algorithm to find out the first time of contact among the features. The model complexity and average CCD time per frame of each benchmark are shown in Table 2.

6.3 Culling Efficiency

Table 3 compares the number of elementary tests related to AVTPs and AETPs with and without the adjacency-based culling method. As shown in the table, the elementary tests relevant to adjacent triangles are dramatically cut down. Table 4 shows the number of elementary tests before and after duplication elimination. For all the benchmarks, approximately 3/4 of elementary tests are duplicated, and their computation can be avoided by querying from the feature test table.

7 Comparison and Analysis

7.1 Comparison

In [8], k-DOPs are also used as bounding volumes of features. By using bounding volume tests prior to elementary tests, parts of false positives can be removed. Wong and Baciu [23] present a feature-based CCD algorithm. By using a randomized marking scheme, features

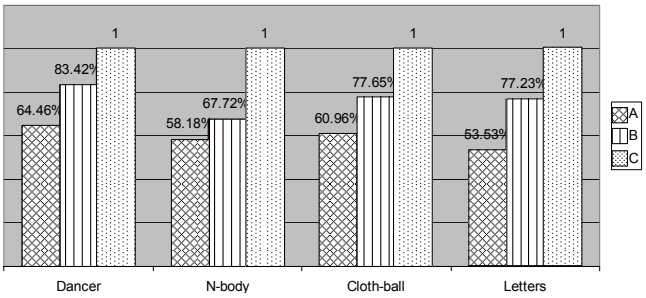

Fig. 6 Efficiency comparison: running time of our method (A), a combination of [8] and [23] (B), and [8] only (C).

are distributed among the triangles whose those feature belongs to. This method can achieve good culling efficiency; the elementary tests are checked for once and only once. Although these methods have their own benefits, both methods may suffer from a high number of false positive elementary pairs that arise from adjacent triangles.

Adjacency-based culling method can directly skip all the adjacent triangle pairs, and the table-based duplication elimination can achieve the same culling efficiency of [23]: all the elementary tests are performed only once, at the price of maintaining the feature test table and searching in the table.

To highlight the benefit of our algorithm, we implemented a combination of the methods of [8] and [23], and compare its running time with our algorithm. Fig. 6 shows the result of the comparison: "A" is an implementation of our method, "B" is an implementation based on the combination of [8] and [23], and "C" is an implementation of the method of [8] only. As shown in the figure, the running time of "A" is about $53\% - 64\%$ of "C", and is about $70\% - 86\%$ of "B".

Our algorithm is independent of the choice of underlying BVH and is complementary to other triangle-based culling methods. As a triangle-based method, it can integrate with the continuous normal cone culling method [19] seamlessly, whereas [23] is hard to integrate with other triangle-based culling methods due to the random distribution of features among triangles. As an extension of [23], [3] has the same characteristic.

In [19], a low-level culling method, called "Orphan test", is used as triangle-level culling. It is equivalent in concept to our adjacency based culling, and its culling efficiency is of the same magnitude (or slightly better) as our method. Both these approaches have comparable performance.

7.2 Limitations

There are some limitations in our methods: The high culling efficiency of our approach heavily relies on the adjacency between triangles. So when the objects break

into pieces, the benefit of adjacency-based culling decreases. Moreover, the use of hash table increases the memory overhead of our approach.

8 Conclusion and Future Work

By utilizing the adjacency between triangles, we present an efficient algorithm for CCD between complex deformable models, including self-collisions. Our algorithm is based on a triangle-based hierarchical culling method named adjacent-based culling and a table-based duplication elimination technique. The algorithm is applicable to rigid and deformable models

As part of future work, we will like to extend this approach to efficiently handle scenarios with breaking or changing topologies. Moreover, we would like to parallelize the algorithms on processors with multiple cores.

Acknowledgements We would like to thank Sean Curtis for many useful discussions and his initial code for collision detection. We thank Stephane Redon for his elementary test codes. We also thank Rasmus Tamstorf, Naga Govindaraju, Avneesh Sud, Russ Gayle and Ming Lin for useful discussions and the benchmarks. This research is supported in part by ARO Contracts DAAD19-02-1-0390 and W911NF-04-1-0088, NSF awards 0400134, 0429583 and 0404088, DARPA/RDECOM Contract N61339-04-C-0043, Disney and Intel, KAIST seed grant, and the IT R&D program of MKE/IITA [2008-F-033-01, Development of Real-time Physics Simulation Engine for e-Entertainment]. Tang is supported in part by National Basic Research Program of China (No. 2006CB303106), Natural Science Foundation of Zhejiang, China (No. Y107403), Doctoral subject special scientific research fund of Education Ministry of China (No. 20070335074), and Future Academic Star fellowship from Zhejiang University.

References

1. Baraff, D., Witkin, A., Kass, M.: Untangling cloth. Proc. of ACM SIGGRAPH pp. 862–870 (2003)
2. van den Bergen, G.: Efficient collision detection of complex deformable models using AABB trees. Journal of Graphics Tools **2**(4), 1–14 (1997)
3. Curtis, S., Tamstorf, R., Manocha, D.: Fast collision detection for deformable models using representative-triangles. In: SI3D '08: Proceedings of the 2008 Symposium on Interactive 3D graphics and games, pp. 61–69 (2008)
4. Foskey, M., Garber, M., Lin, M., Manocha, D.: A voronoi-based hybrid planner. Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (2001)
5. Gottschalk, S., Lin, M., Manocha, D.: OBB-Tree: A hierarchical structure for rapid interference detection. Proc. of ACM Siggraph'96 pp. 171–180 (1996)
6. Govindaraju, N., Knott, D., Jain, N., Kabul, I., Tamstorf, R., Gayle, R., Lin, M., Manocha, D.: Collision detection between deformable models using chromatic decomposition. ACM Trans. on Graphics (Proc. of ACM SIGGRAPH) **24**(3), 991–999 (2005)
7. Hoff, K., Culver, T., Keyser, J., Lin, M., Manocha, D.: Interactive motion planning using hardware accelerated computation of generalized voronoi diagrams. Proceedings of IEEE Conference of Robotics and Automation (2000)
8. Hutter, M., Fuhrmann, A.: Optimized continuous collision detection for deformable triangle meshes. In: Proc. WSCG '07, pp. 25–32 (2007)
9. Klosowski, J., Held, M., Mitchell, J., Sowizral, H., Zikan, K.: Efficient collision detection using bounding volume hierarchies of k-dops. IEEE Trans. on Visualization and Computer Graphics **4**(1), 21–37 (1998)
10. Larsson, T., Akenine-Möller, T.: A dynamic bounding volume hierarchy for generalized collision detection. Computers and Graphics **30**(3), 451–460 (2006)
11. Lauterbach, C., Yoon, S., Tuft, D., Manocha, D.: RT-DEFORM: Interactive Ray Tracing of Dynamic Scenes using BVHs. IEEE Symposium on Interactive Ray Tracing pp. 39–46 (2006)
12. LaValle, S.M.: Planning Algorithms. Cambridge University Press (also available at http://msl.cs.uiuc.edu/planning/) (2006)
13. Lin, M., Manocha, D.: Collision and proximity queries. In: Handbook of Discrete and Computational Geometry (2003)
14. Pisula, C., Hoff, K., Lin, M., Manocha, D.: Randomized path planning for a rigid body based on hardware accelerated voronoi sampling. In: Proc. of 4th International Workshop on Algorithmic Foundations of Robotics (2000)
15. Provot, X.: Collision and self-collision handling in cloth model dedicated to design garment. Graphics Interface pp. 177–189 (1997)
16. Redon, S., Kheddar, A., Coquillart, S.: Fast continuous collision detection between rigid bodies. Proc. of Eurographics (Computer Graphics Forum) **21**(3), 279–288 (2002)
17. Redon, S., Kim, Y.J., Lin, M.C., Manocha, D.: Fast continuous collision detection for articulated models. In: Proceedings of ACM Symposium on Solid Modeling and Applications, pp. 145–156 (2004)
18. Sud, A., Otaduy, M.A., Manocha, D.: DiFi: Fast 3D distance field computation using graphics hardware. Computer Graphics Forum (Proc. Eurographics) **23**(3), 557–566 (2004)
19. Tang, M., Curtis, S., Yoon, S., Manocha, D.: Interactive continuous collision detection between deformable models using connectivity-based culling. Proc. of SPM08 (ACM Solid and Physical Modeling Symposium) (2008)
20. Teschner, M., Kimmerle, S., Heidelberger, B., Zachmann, G., Raghupathi, L., Fuhrmann, A., Cani, M.P., Faure, F., Magnenat-Thalmann, N., Strasser, W., Volino, P.: Collision detection for deformable objects. Computer Graphics Forum **19**(1), 61–81 (2005)
21. Volino, P., Thalmann, N.M.: Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. Computer Graphics Forum (EuroGraphics Proc.) **13**(3), 155–166 (1994)
22. Wong, W.S.K., Baciu, G.: Dynamic interaction between deformable surfaces and nonsmooth objects. IEEE Tran. on Visualization and Computer Graphics **11**(3), 329–340 (2005)
23. Wong, W.S.K., Baciu, G.: A randomized marking scheme for continuous collision detection in simulation of deformable surfaces. Proc. of ACM VRCIA pp. 181–188 (2006)
24. Yoon, S., Curtis, S., Manocha, D.: Ray tracing dynamic scenes using selective restructuring. Proc. of Eurographics Symposium on Rendering (2007)
25. Zhang, L., Manocha, D.: Motion interpolation with distance constraints. Tech. Rep. TR 08-001, Department of Computer Science, UNC Chapel Hill (2008)
26. Zhang, X., Redon, S., Lee, M., Kim, Y.J.: Continuous collision detection for articulated models using taylor models and temporal culling. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) **26**(3), 15 (2007)

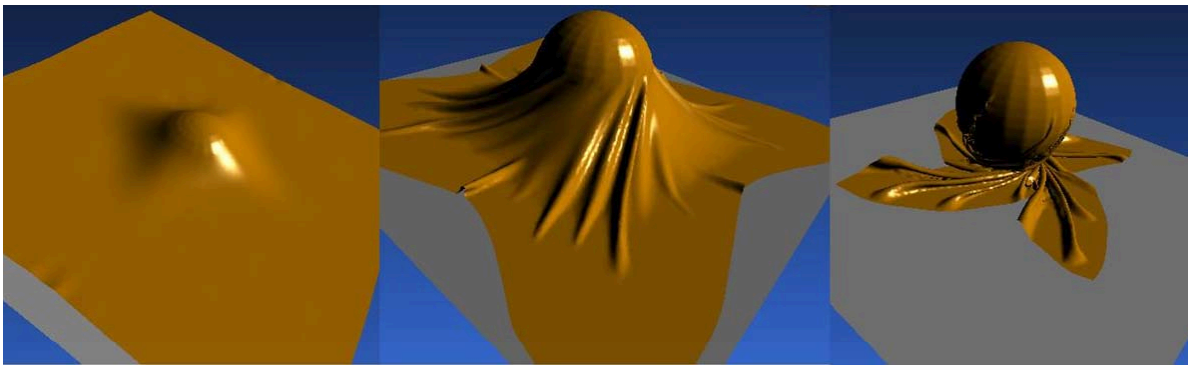

Fig. 7 Cloth simulation benchmark: “Cloth-ball”.

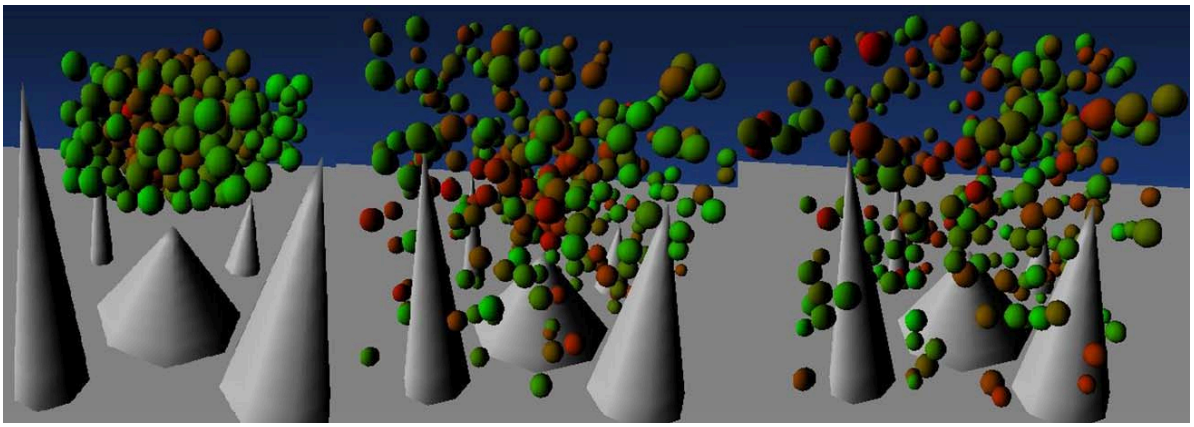

Fig. 8 N-body simulation benchmark: “N-body”.

Fig. 9 Cloth simulation benchmark: “Dancer”.

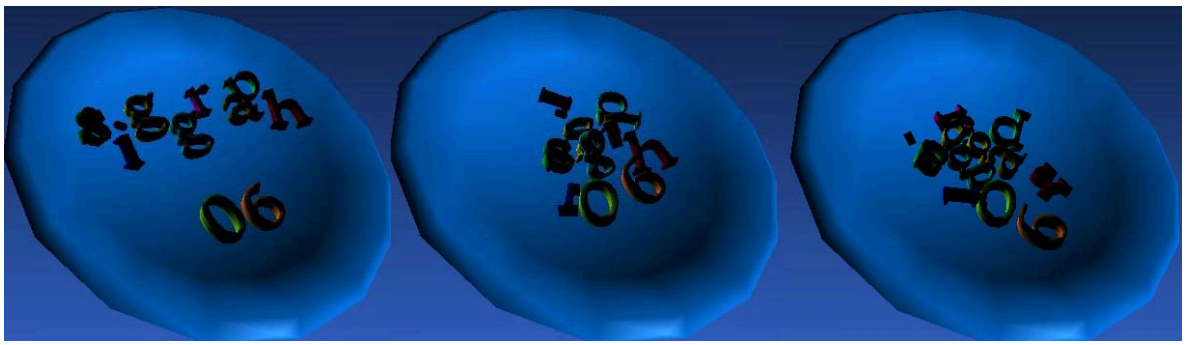

Fig. 10 N-body simulation benchmark: “Letters”.