Fast Continuous Collision Culling with Deforming Non-collinear Filters

Peng Du
Zhejiang University
dp@zju.edu.cn

Min Tang[*]
Zhejiang University
tang_m@zju.edu.cn

Ruofeng Tong
Zhejiang University
trf@zju.edu.cn

Abstract

We present a novel culling algorithm that uses deforming non-collinear filters to improve the performance of continuous collision detection (CCD) algorithms. The underlying idea is to use simple and effective filters, deforming non-collinear filters (NCFs), that reduce the number of false positives between the primitives. These filters are derived from the collinear conditions and can be easily combined with other culling methods. We have tested its performance on several benchmarks. Compared with previous methods, we can reduce the number of false positives significantly and improve the overall performance of CCD algorithms, especially for simulations with large time steps.

Keywords: continuous collision detection, deforming non-collinear filters, bounding volume hierarchies, deformable models

1 Introduction

Continuous collision detection (CCD) is widely used in physically-based simulation, CAD/CAM, and robot motion planning [1, 2]. It can be used to compute the first-time-of-contact between deforming triangles along continuous trajectories. The CCD test between two deforming triangles reduces to performing 9 vertex-face (VF) and 6 edge-edge (EE) elementary tests. Each elementary test need to compute the roots of a cubic equation.

[*]Corresponding author. Email:tang_m@zju.edu.cn.

Main Results: We present a novel culling algorithm that can significantly reduce the number of false positives in terms of elementary tests and improve the overall performance of CCD algorithms. We introduce new deforming non-collinear filters (NCFs) which can remove many false positives effectively. The main idea is to exploit the collinear condition of the elementary tests along the continuous deforming trajectory. The NCFs can be used for culling of VF tests and EE tests.

Our algorithm is complementary to prior CCD algorithms and can be easily combined with prior high-level and low-level culling algorithms. We have tested its performance on several complex benchmarks. The experiment results show that the algorithm is well suited for the deformable models with many self-collisions and can cull up to 99% of false positives.

Organization: The rest of the paper is organized as follows: Section 2 gives a brief survey of related works. The notations and theorems are introduced in Section 3. The deforming non-collinear filter and overall CCD algorithm will be described in Section 4. We compare our method with prior algorithms in Section 5.

2 Related Works

Many efficient culling algorithms have been designed for CCD between rigid models and deformable models. Their main purposes are to reduce the number of the elementary tests, since the exact VF and EE tests need to solve cubic equations. Most of these culling algorithms exploit BVHs as the basic culling method, and combine with other high-level and low-level culling methods.

2.1 High-level Culling

High-level culling performs triangle-level overlap tests to reduce the potential collided pairwise set (PCS). Bounding volume hierarchies (BVHs) have been widely used as high-level culling methods. Some widely used bounding volumes include spheres [3, 4], axis-aligned bounding boxes (AABBs) [5], oriented bounding boxes (OBBs) [6], discretely oriented polytopes (k-DOPs) [7], etc. In addition, several culling methods aim at improving the culling efficiency of self-collision detection, including continuous normal cone tests [8], star-contour tests [9], and subspace min-norm certification tests [10]. Recently, some algorithms that exploit the parallel computing capability of modern multi-core CPUs or many-core GPUs have been designed [11, 12, 13, 14].

2.2 Low-level Culling

Low-level culling is to remove redundant primitive pairs (VF or EE pair) and reduce elementary tests from PCS. Recently, representative triangles [15] and orphan sets [8] are proposed to remove redundant primitive pairs. Continuous separating axis theorem [16], deforming non-penetration filter [17] and parallel filter in subspace [18] are proposed to reduce a large number of false positives. Even with all these culling methods, the current CCD algorithms still result in a high number of false positives.

3 Notations and Motivation

In this section, we introduce the notations used and illustrate our motivation of the deforming non-collinear filters (NCFs).

3.1 Notations

- Vertex, edge, and triangle are represented by the symbols P, E and T, respectively. We perform CCD at the time interval where $t \in [0, 1]$.

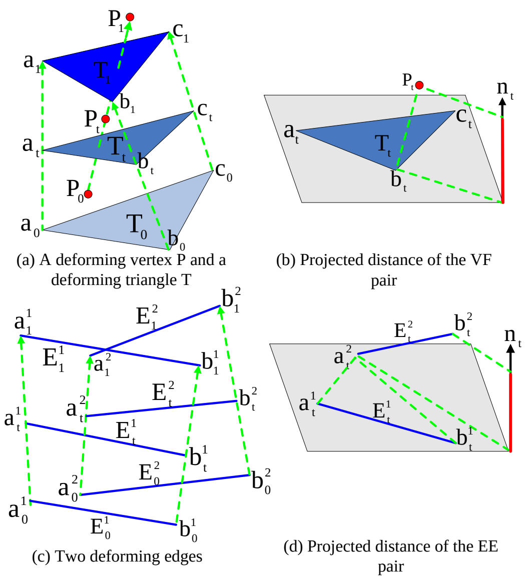

(a) A deforming vertex P and a deforming triangle T
(b) Projected distance of the VF pair
(c) Two deforming edges
(d) Projected distance of the EE pair

Figure 1: **Deforming non-penetration filters:** For a VF pair, if a vertex P is always on the same side of a triangle T along the entire linear trajectory, this pair should be culled. For an EE pair, if there is no crossing between the two deforming edges along the linear trajectory, this pair should be culled.

- The position of each vertex between time steps, is defined by a linear interpolation function $F(t)$. The positions of a specific vertex, edge and triangle are denoted by symbols P_t, E_t, T_t for a certain time t where $t \in [0, 1]$.
- Symbols a_t, b_t, c_t are used to denote the corresponding vertices of edges and triangles at time t. $a_t b_t$, $b_t c_t$, $c_t a_t$ are used to denote the corresponding edges of the triangle. n_t is the normal vector of T_t.
- Operator $*$, $\cdot$, $\times$ denote multiplication of a scalar and a vector, dot product of two vectors, and cross product of two vectors, respectively.

3.2 Motivation

The CCD test between a triangle pair can be reduced to two types of elementary tests: 6 VF tests and 9 EE tests. Each elementary test can be further broken down into two parts: a coplanar test and an inside test. Both the VF tests and EE tests involve the use of four deforming vertices, and a necessary condition for a collision is that these four vertices are coplanar. Given the linear interpolating motion between the vertices, the coplanar test of four vertices can be reduced to find roots of a cubic equation [19]. We use the Interval-Newton method to solve the cubic equations [20], which takes 155 additions, 217 multiplications, and 6 divisions on average (float-point operations). An inside test which determines whether the colliding point is inside a triangle (for VF tests) or inside both of the edges (for EE tests), takes 9 additions, 28 multiplications, and 3 divisions on average.

The cost for elementary tests is very expensive. Therefore, if the triangle pairs cannot be culled by the bounding volume tests, we make use of deforming non-penetration filter (DNF) [17] for further culling: as shown in Fig. 1(b), if vertex P is always on one side of triangle T along the entire linear trajectory, this pair should be classified as a false positive since it cannot be coplanar. It is a sufficient condition to check whether P_t will always be on the same side of T_t during the time interval $[0, 1]$. If a pair of primitives satisfies this condition, then it doesn't need to perform the exact test in terms of solving a cubic equation. The EE tests have the same feature: if there is no crossing between the two deforming edges along the linear trajectory, they cannot be coplanar during the time interval (Fig. 1(d)). However, DNF can become quite conservative for deformable models with large time steps. Therefore, we propose a deforming non-collinear filter to solve this problem.

We make use of the following observation: as shown in Fig. 2(a), for VF test, in the orthogonal plane perpendicular to n_t, if vertex P_t is always on one side of an edge adjacent to the triangle, and the triangle vertex not adjacent to the edge is always on the other side of the edge, this pair should be classified as a false positive. The idea can be extended for EE test: in the orthogonal plane perpendicular to n_t, if the two vertices of one edge are always on the same side of the other edge during the time interval $[0, 1]$, this pair should be classified as a false positive (Fig. 2(b)). The above method can be summarized as:

Theorem 1 (Projection Culling Theorem) *At any time interval, for some orthogonal projection, if there is no contact for two deforming primitives in the projection space, there is no contact between the two primitives in their original space.*

Proof. Assume that there is a cross between two deforming primitives in their original space, there must have a contact between the two deforming primitives in the orthogonal projection plane, which is contrary to the assumption of Theorem 1.

4 Deforming Non-collinear Filter

In this section, we present our culling algorithm based on the Projection Culling Theorem. The objective is to derive sufficient conditions for non-overlap elements during the time interval. We first present the collinear formulation for a vertex P and an edge ab undergoing continuous motion. Next, we extend it to perform VF and EE culling tests.

4.1 Non-collinear Test

In order to check the collinearity of P and ab, we need to calculate the projected distance between P_t and n_p, as shown in Fig. 2(a). If this distance becomes zero at any time interval, then the pair of the two primitives are classified as collinearity. This method can be summarized as:

Theorem 2 (Non-collinear Theorem) *For an edge $a_t b_t$ and a vertex P_t defined by the start and end positions during the interval $[0, 1]$, these positions are linearly interpolated in the interval with respect to the time variable. If the following five scalar values: $(P_0 - b_0) \cdot A$, $(P_0 - b_0) \cdot (C+F)+(P_1-b_1)\cdot A$, $(P_0-b_0)\cdot(D+E)+(P_1-b_1) \cdot (C+F)$, $(P_0-b_0) \cdot B+(P_1-b_1) \cdot (D+E)$ and $(P_1-b_1) \cdot B$ have the same sign, P_t and $a_t b_t$ will not be collinear during the interval, where*

$$A = (b_0 - a_0) \times n_0, B = (b_1 - a_1) \times n_1,$$

$$C = (b_0 - a_0) \times \hat{n}, D = (b_1 - a_1) \times \hat{n},$$

$$E = (b_0 - a_0) \times n_1, F = (b_1 - a_1) \times n_0,$$

and:

$$n_0 = (b_0 - a_0) \times (c_0 - a_0),$$

$$n_1 = (b_1 - a_1) \times (c_1 - a_1),$$

$$\hat{n} = n_0 + n_1 - (\vec{v_b} - \vec{v_a}) \times (\vec{v_c} - \vec{v_a}),$$
$$\vec{v_a} = a_1 - a_0, \vec{v_b} = b_1 - b_0, \vec{v_c} = c_1 - c_0.$$

Proof. As shown in Fig. 2(a), the normal vector n_t of the deforming triangle at any time instance t can be represented as follows:

$$n_t = n_0 * B_0^2(t) + \hat{n} * 1/2 * B_1^2(t) + n_1 * B_2^2(t) \quad (1)$$

where B_i^2 is the i^{th} basis function of the Bernstein polynomials of degree 2.

For the vertices of the deforming triangle, we make $a_t = a_0 * (1-t) + a_1 * t$ and $b_t = b_0 * (1-t) + b_1 * t$. In the orthogonal plane perpendicular to n_t, the normal vector n_p of edge $a_t b_t$ can be represented as follows:

$$\begin{aligned} n_p &= (b_t - a_t) \times n_t \\ &= ((b_0 - a_0) * (1-t) + (b_1 - a_1) * t) \times n_t \\ &= A * B_0^3(t) + (C+F) * 1/3 * B_1^3(t) + \\ &\quad (D+E) * 1/3 * B_2^3(t) + B * B_3^3(t) \quad (2) \end{aligned}$$

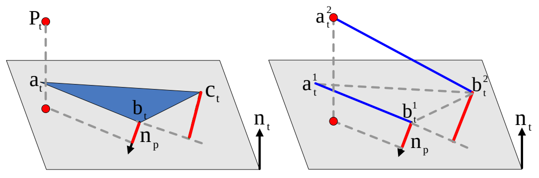
where B_i^3 is the i^{th} basis function of the Bernstein polynomials of degree 3.

For the moving vertex $P_t = P_0 * (1-t) + P_1 * t$, its projected distant along n_p is:

$$\begin{aligned} & (P_t - b_t) \cdot n_p \\ = & ((P_0 - b_0) * (1-t) + (P_1 - b_1) * t) \cdot n_p \\ = & (P_0 - b_0) \cdot A * B_0^3 * (1-t) + \\ & (P_0 - b_0) \cdot (C+F) * 1/3 * B_1^3 * (1-t) + \\ & (P_0 - b_0) \cdot (D+E) * 1/3 * B_2^3 * (1-t) + \\ & (P_0 - b_0) \cdot B * B_3^3 * (1-t) + \\ & (P_1 - b_1) \cdot A * B_0^3 * t + \\ & (P_1 - b_1) \cdot (C+F) * 1/3 * B_1^3 * t + \\ & (P_1 - b_1) \cdot (D+E) * 1/3 * B_2^3 * t + \\ & (P_1 - b_1) \cdot B * B_3^3 * t \quad (3) \end{aligned}$$

After reduction, equation (3) can be represented as follows:

$$\begin{aligned} & (P_t - b_t) \cdot n_p = (P_0 - b_0) \cdot A * B_0^4 \\ & + \frac{(P_0 - b_0) \cdot (C+F) + (P_1 - b_1) \cdot A}{4} * B_1^4 \\ & + \frac{(P_0 - b_0) \cdot (D+E)}{6} * B_2^4 \\ & + \frac{(P_1 - b_1) \cdot (C+F)}{6} * B_2^4 \\ & + \frac{(P_0 - b_0) \cdot B + (P_1 - b_1) \cdot (D+E)}{4} * B_3^4 \\ & + (P_1 - b_1) \cdot B * B_4^4 \quad (4) \end{aligned}$$



(a) Deforming non-collinear filter for VF pair

(b) Deforming non-collinear filter for EE pair

Figure 2: Deforming non-collinear filter: for VF pair, in the orthogonal plane perpendicular to n_t, if vertex P_t is always on one side of edge $a_t b_t$, and vertex c_t is always on the other side of $a_t b_t$, this pair should be culled out; for EE pair, in the orthogonal plane perpendicular to n_t, if the two vertices of one edge are always on the same side of the other edge during the time interval $[0, 1]$, this pair should be culled out.

where B_i^4 is the i^{th} basis function of the Bernstein polynomials of degree 4. If the five coefficients have the same sign, P_t cannot be collinear with E_t depending on the convex hull property associated with control points of the Bernstein basis.

4.2 Primitive Culling

Non-collinear Theorem for VF Test: The VF filter is based on the extension of the Non-collinear Theorem.

Corollary 1 (VF Filter) *In the orthogonal plane of n_t, if the projection of vertex P_t is always on one side of edge $a_t b_t$, and vertex c_t is always on the other side of the edge, this pair should be classified as a false positive (Fig. 2(a)).*

We do filter culling process in the orthogonal plane of n_t. Firstly we decide whether the triangle will turn over during the time interval by selecting one vertex and its opposite edge from the triangle and testing whether the vertex and its opposite edge are collinear with Theorem 2. If the triangle never turns over, we continue to figure out the cutting edge by comparing the sign of the five scalar values between P_t and each edge of T_t, which has been introduced in the Non-collinear Theorem. The cutting edge separates vertex P_t and triangle T_t in the time interval [0, 1]. We select the first one we find. The whole

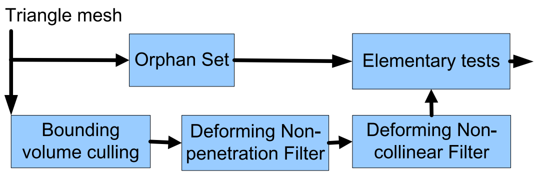

Triangle mesh
Orphan Set
Elementary tests
Bounding volume culling
Deforming Non-penetration Filter
Deforming Non-collinear Filter

Figure 3: Process of our algorithm: In the initial work, we construct BVH. Then we use Orphan Set [8] to get the adjacent collision pairs. Finally, we use bounding volume tests, deforming non-collinear filter, deforming non-collinear filter and exact elementary test sequentially to get the non-adjacent collision pairs.

process is summarized as Corollary 1 and illustrated in Algorithm 1.

Non-collinear Theorem for EE Test: The Non-collinear Theorem also can be extended for the EE tests. The whole process is summarized as Corollary 2 and illustrated in Algorithm 2.

Corollary 2 (EE Filter) *In the orthogonal plane of n_t, if the two vertices a_t^2, b_t^2 of one edge are always on the same side of the other edge $a_t^1 b_t^1$ during the time interval $[0,1]$, this pair should be classified as a false positive (Fig. 2(b)).*

4.3 CCD Algorithm

We use the non-collinear filter defined above to perform collinearity based culling. Our filter can be combined with hierarchical representations (Fig. 3). After the bounding volume tests, we perform the non-collinear filter to remove redundant primitive pairs that are not in close proximity to one another. Next, the deforming non-collinear filter is used as part of low-level culling to further remove the false positives.

5 Implementation and Results

In this section, we describe our implementation and compare the performance of our algorithm against previous methods on several benchmarks.

Implementation: We have implemented our algorithm on a standard 2.33GHz Intel Pentium machine with 3.75GB RAM on a 32-bit Windows 7 platform. The performance is measured using a single thread. k-DOPs (specifically 16-DOPs) are used as bounding volumes because they provide a good balance between tight fitting and rapid updating. We use BVH refitting to update the hierarchy for deformable models. Orphan set [8] and representative triangle (R-Triangle) [15] are employed to reduce the redundant elementary tests. We use deforming non-penetration filter (DNF) [17] to cull the false positives followed by deforming non-collinear filter (NCF).

Algorithm 1 Non-collinear Theorem for VF Test: For a given moving vertex P_t and a given moving triangle T_t, we decide whether this primitive pair can be culled out. $M_1 \sim M_5$ are used to denote the five scalar values between a_t and $b_t c_t$ in the Non-collinear Theorem. $N_1 \sim N_5$ are used to denote the five scalar values between P_t and an edge e_t of the triangle in the Non-collinear Theorem.

```
1: if $M_1 \sim M_5$ have different signs then
2:     return false //The triangle turns over.
3: for all $e_t = b_t c_t, c_t a_t, a_t b_t$ do
4:     if $N_1 \sim N_5$ have the same sign, and they have different signs with $M_1$ then
5:         return true //Can be culled.
6: return false //Cannot be culled.
```

Benchmarks & Performance: In order to test the performance of our algorithm, we used four different benchmarks as follows:

- Airbag (18.2K triangles): The deforming airbag has intra-object collisions as well as inter-object collisions with the steering wheel (Fig. 6(a)).

- Cloth-ball (92K triangles): A piece of cloth drops on top of a ball and curls around resulting in a high number of inter- and intra-object collisions (Fig. 6(b)).

- Funnel (92K triangles): A cloth falls into a funnel and pass through it under the pressure of a ball. This model has a lot of inter- and intra-object collisions (Fig. 6(c)).

- Bunny (4K triangles): A poor bunny is smashed onto the ground by a steel plate. This benchmark has a high number of inter- and intra-object collisions (Fig. 6(d)).

Algorithm 2 Non-collinear Theorem for EE Test: For two moving edges, we decide whether this primitive pair can be culled out. $M_1 \sim M_5$ and $N_1 \sim N_5$ are used to denote the five scalar values for $\{a_t^j, a_t^i b_t^i\}$ and $\{b_t^j, a_t^i b_t^i\}$ (where $i = 1, 2$, and $j = 3 - i$) in the Non-collinear Theorem, respectively.

```
1: Get $M_1 \sim M_5$ for $\{a_t^2, a_t^1 b_t^1\}$
2: if $M_1 \sim M_5$ have the same sign then
3:     Get $N_1 \sim N_5$ for $\{b_t^2, a_t^1 b_t^1\}$
4:     if $N_1 \sim N_5$ have the same sign then
5:         return true //Can be culled.
6: Get $M_1 \sim M_5$ for $\{a_t^1, a_t^2 b_t^2\}$
7: if $M_1 \sim M_5$ have the same sign then
8:     Get $N_1 \sim N_5$ for $\{b_t^1, a_t^2 b_t^2\}$
9:     if $N_1 \sim N_5$ have the same sign then
10:        return true //Can be culled.
11: return false //Cannot be culled.
```

Fig. 4 highlights the culling efficiency of our algorithm by comparing the number of elementary tests performed by our method, DNF, separating axis theorem (SAT) [16], R-Triangle and parallel filter in subspace (PFS) [18]. In addition, we have tested the culling rate of NCF after we perform the culling operation of DNF. As shown in Fig. 5, for the non-culled primitives of DNF, we can get a more than 90% culling rate with NCF. The average running time per frame in SAT, R-Triangle and our method is shown in Tab. 1. Experiment results prove, we can get better performance especially for large time steps.

Analysis and Comparison: The principle of filter culling is to use low costly culling tests to replace the high costly elementary tests in order to reduce the executed time. In addition, the time reduction is determined by the culling rate and the culling cost. The larger the culling rate and the smaller the culling cost is, the more remarkable the acceleration effect appears [18].

In our method, we use DNF to cull part of false positives which takes about 29 additions and 40 multiplications. Then we use the non-collinear filter for further culling which takes about 35 additions and 80 multiplications. Finally, we use the Interval-Newton method to solve the cubic equations for the elementary test. The elementary test takes about 164 additions, 245 multiplications, and 9 divisions on average

Table 1: Performance Comparison on Executed Time

Models	Our method (ms/frame)	Speedups over SAT	Speedups over R-Triangle
Airbag 1×[*]	150	1.16×	1.41×
Airbag 3×	330	1.32×	1.37×
Cloth-ball 1×	938	1.24×	1.53×
Cloth-ball 3×	3507	1.36×	1.28×
Funnel 1×	73	1.09×	2.16×
Funnel 3×	139	1.14×	1.64×
Bunny 1×	12	1.54×	2.36×
Bunny 3×	38	1.67×	1.89×

[*] The time step, 1× denotes the original time interval. 3× means the time step is enlarged by 3 times.

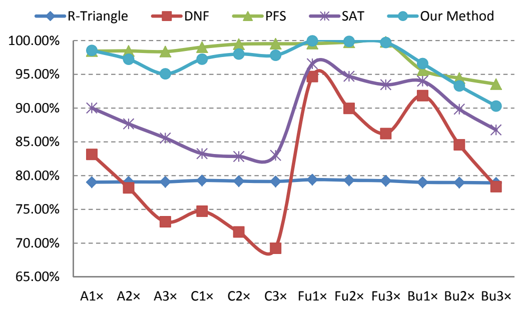


Figure 4: Culling efficiency: A, C, Fu and Bu denotes airbag, cloth-ball, funnel and bunny models respectively. n× denotes n times the length of the original time interval.

(including the coplanar and inside tests). From the benchmarks, we see our method is a good complementary for DNF to improve the culling efficiency.

We have implemented some previous methods (including DNF, PFS, SAT and R-Triangle). In the following section, we would like to compare our method with these culling methods.

- **Parallel Filter in Subspace:** They apply a parallel linear filter and parallel planar filter in eight different projection subspaces with SIMD capacity [13]. As shown in Fig. 4, they can get the best culling efficiency and time reduction. However, their efficiency comes from 8 projection axes and they use SIMD capacity to parallel the filter in the 8 axes. It makes this method cannot be transplanted to GPU directly, while our method can avoid the problem.

- **Separating Axis Theorem:** Given two elements, the separating axis theorem (SAT) [16] states that if there exists a line onto

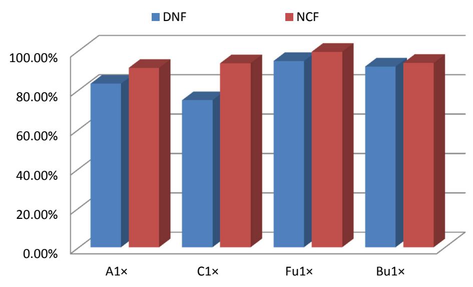


Figure 5: The culling rate of NCF following DNF: We have calculated the culling efficiency of NCF after DNF culling. For the non-culled primitives of DNF, we can get a more than 90% culling rate with NCF.

which the projections of two elements do not overlap, then the elements do not intersect. Experiments prove that our method has better time reduction and higher culling efficiency than SAT on the four benchmarks.

- **Representative Triangles:** R-Triangle [15] can remove all the redundant elementary tests that are caused by shared features between the triangles, which has the culling rate of less than 80%.

- **Deforming Non-penetration Filter:** DNF [17] uses coplanar condition to cull the false positives. For the benchmarks owning many self-collisions and large time steps, its culling efficiency is rather poor.

Limitation: Our approach only provides a filter at the feature level. When the culling efficiency of DNF is high (above 95% in our benchmarks), our method is less efficient. Meanwhile, our method is not efficient for rigid-body collision detection. If a high-level culling algorithm is able to cull away a high percentage of false positives, we may not obtain an improvement with deforming non-collinear filter.

6 Conclusion

We have presented a novel culling algorithm for CCD between complex deformable models by proposing deformable non-collinear filter. By combining with DNF, our algorithm can significantly reduce the number of false positives, and

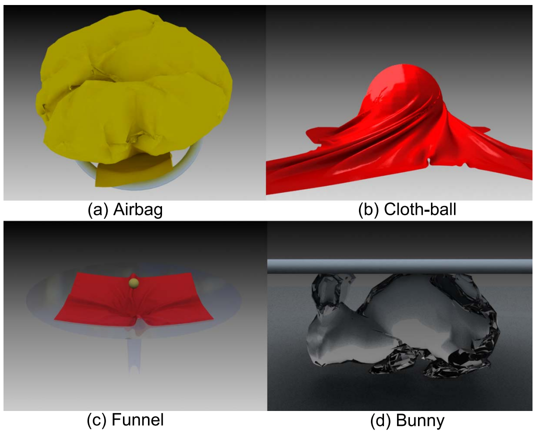


Figure 6: Benchmarks: all the benchmarks have multiple simulation steps with many inter- and intra-object collisions.

subsequently improve the overall performance of CCD. Moreover, our method is easy to combine with almost every prior collision detection methods based on BVHs. We have tested the performance on different benchmarks and observed considerable improvement in terms of reducing the number of false positives.

In our future work, it is interesting to improve our filter that would make it efficient for rigid-body collision detection. We also would like to integrate our CCD algorithm with some well known game physics engines, such as Bullet and PhysX.

Acknowledgements

The project is supported in part by National Basic Research Program (2011CB302205), National Key Technology R&D Program (2012BAD35B01), Natural Science Foundation of China (61170140), Natural Science Foundation of Zhejiang, China (Y1100069, Y1100018).

References

[1] M. Chen and K. Tang. A fully geometric approach for developable cloth deformation simulation. *The Visual Computer*, 26:853–863, 2010.

[2] G. Böttcher, D. Allerkamp, and F. E. Wolter. Multi-rate coupling of physical

simulations for haptic interaction with deformable objects. *The Visual Computer*, 26:903–914, 2010.

[3] P. M. Hubbard. Interactive collision detection. In *Proceedings of IEEE Symposium on Research Frontiers in Virtual Reality*, pages 24–31, 1993.

[4] G. Bradshaw and C. O'Sullivan. Adaptive medial-axis approximation for sphere-tree construction. *ACM Transactions on Graphics*, 23:1–26, 2004.

[5] G. V. D. Bergen. Efficient collision detection of complex deformable models using aabb trees. *Journal of Graphics Tools*, 2:1–14, 1998.

[6] S. Gottschalk, M. Lin, and D. Manocha. Obbtree: A hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 171–180, 1996.

[7] J. Klosowski, M. Held, J. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Transactions on Visualization and Computer Graphics*, 4:21–37, 1998.

[8] M. Tang, S. Curtis, S. E. Yoon, and D. Manocha. Interactive continuous collision detection between deformable models using connectivity-based culling. In *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pages 25–36, 2008.

[9] S. C. Schvartzman, Á. G. Pérez, and M. A. Otaduy. Star-contours for efficient hierarchical self-collision detection. *ACM Transactions on Graphics*, 29:801–808, 2010.

[10] J. Barbič and D. L. James. Subspace self-collision culling. *ACM Transactions on Graphics*, 29:811–819, 2010.

[11] D. Kim, J. P. Heo, J. Huh, J. Kim, and S. E. Yoon. HPCCD: Hybrid parallel continuous collision detection using CPUs and GPUs. *Computer Graphics Forum*, 28:1791–1800, 2009.

[12] S. Pabst, A. Koch, and W. Straßer. Fast and scalable CPU/GPU collision detection for rigid and deformable surfaces. *Computer Graphics Forum*, 29:1605–1612, 2010.

[13] M. Tang, D. Manocha, and R. F. Tong. MCCD: Multi-core collision detection between deformable models using front-based decomposition. *Graphical Models*, 72:7–23, 2010.

[14] M. Tang, D. Manocha, J. Lin, and R. F. Tong. Collision-streams: fast GPU-based collision detection for deformable models. In *Proceedings of the ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pages 63–70, 2011.

[15] S. Curtis, R. Tamstorf, and D. Manocha. Fast collision detection for deformable models using representative-triangles. In *Proceedings of the symposium on Interactive 3D graphics and games*, pages 61–69, 2008.

[16] M. Tang, D. Manocha, S. E. Yoon, P. Du, J. P. Heo, and R. F. Tong. VolCCD: Fast continuous collision culling between deforming volume meshes. *ACM Transactions on Graphics*, 30:111–125, 2011.

[17] M. Tang, D. Manocha, and R. F. Tong. Fast continuous collision detection using deforming non-penetration filters. In *Proceedings of the ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pages 7–14, 2010.

[18] C. Tang, S. Li, and G. Wang. Fast continuous collision detection using parallel filter in subspace. In *Proceedings of the ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pages 71–80, 2011.

[19] X. Provot. Collision and self-collision handling in cloth model dedicated to design garment. *Graphics Interface*, 97:177–189, 1997.

[20] S. Redon, A. Kheddar, and S. Coquillart. Fast continuous collision detection between rigid bodies. *Computer Graphics Forum*, 21:279–288, 2002.