Accurate Self-Collision Detection Using Enhanced Dual-Cone Method

Tongtong Wang[a], Min Tang[a,b], Zhendong Wang[a], Ruofeng Tong[a]

http://min-tang.github.io/home/DCC/

[a]*Zhejiang University, China*
[b]*Alibaba-Zhejiang University Joint Institute of Frontier Technologies, China*

Abstract

We present an accurate and robust algorithm for self-collision detection in deformable models. Our method is based on the normal cone test and is suitable for both discrete and continuous collision queries on triangular meshes. We propose a novel means of employing surface normal cones and binormal cones to perform the normal cone test. Moreover, we combine our culling criteria with bounding volume hierarchies (BVHs) and present a hierarchical traversal scheme. Unlike the previous BVH-based dual-cone method, our method can reliably detect all self-collisions, and it achieves appreciable speedup over other high-level culling methods.

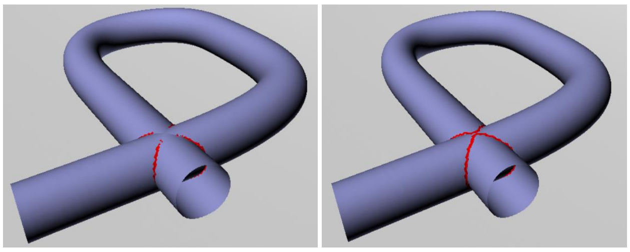
Keywords: Self-Collision Detection, Dual-Cone Culling, BVH

1. Introduction

To ensure the generation of physically plausible results, collision detection (CD) algorithms have been widely used in various applications, including physically based simulations, computer-aided design and computer-aided manufacturing (CAD/-CAM), and robot motion planning. Such algorithms can be classified as either self-collision detection (SCD) for a single object or inter-collision detection among multiple objects. A *false negative* occurs when a CD algorithm misses a collision; a *false positive* occurs when a CD algorithm conservatively classifies a non-collision instance as a collision. An accurate CD method should not result in any *false negatives*.

Most CD algorithms use bounding volume hierarchies (BVHs) for acceleration. These methods work well for inter-object CD, but they incur high computation times for SCD for deformable objects because the adjacent primitives of a deforming mesh are in close proximity and cannot be culled through bounding volume tests. Even if a mesh (with n triangles) has no self-intersection, checking for self-collision is still quite expensive ($O(n^2)$ complexity).

Many approaches have been proposed to improve the efficiency of SCD. Volino and Thalmann [1] introduced the normal cone test for discrete collision detection (DCD). This approach was extended to continuous collision detection (CCD) by Tang et al. [2], leading to more efficient execution of self-intersection queries. Heo et al. [3] proposed a dual-cone culling method based on the normal cone test, which has lower computational overhead but may result in false negatives in practice. To address this problem, they proposed an extension that includes internal boundary edges in [3]. However, although this method



(a) Dual-Cone Method **(b) Our Method**

Figure 1: **Pipe Benchmark.** We illustrate the benefits of our SCD algorithm using the Pipe benchmark (78K triangles). The colliding triangle pairs are highlighted in red. Unlike the previous dual-cone method (a), our method (b) can detect all the collisions.

results in no false negatives, maintaining such internal boundary edges can significantly reduce the performance.

Main Results: In this paper, we propose a new method that not only does not miss collisions but also accelerates the performance of the extension of the original dual-cone method. First, we introduce a sufficient set of criteria for determining whether a surface exhibits self-collisions based on two types of cones and the boundary contours of four sub-surfaces making up the entire surface (Figure 5). The two cone types are surface normal cones and binormal cones. Second, we design a BVH-based hierarchical culling method for use in combination with our culling criteria and present a new bounding volume test tree (BVTT) traversal scheme for our culling criteria, which can significantly reduce the number of redundant tests performed. We evaluate the accuracy of our method on many complex benchmarks involving deformable models and cloth. Unlike the previous dual-cone method [3], our method can accurately detect all self-collisions. Moreover, we observe considerable speedup compared with other SCD methods.

Email addresses: `wtt923@zju.edu.cn` (Tongtong Wang), `tang_m@zju.edu.cn` (Min Tang), `wangzhendong@zju.edu.cn` (Zhendong Wang), `trf@zju.edu.cn` (Ruofeng Tong)

2. Related Work

In this section, we present a brief review of previous works on CD.

High-level culling: The simplest culling algorithms compute geometric bounds and use BVHs to accelerate CD. Many alternative culling methods have been proposed to reduce the number of queries. Volino and Thalmann [1] proposed the normal cone test for DCD, which takes advantage of the topology and connectivity of the input mesh and checks for self-collision by means of normal cones and 2D contour tests. Many self-collision culling techniques [4, 5, 2, 3, 6] have been developed based on the normal cone test. In addition, Barbič and James [7] presented a self-collision culling method for subspace deformable models, but their method does not support general deformations. Based on this method, Zheng and James [8] proposed an energy-based culling method that is applicable to general deformable models. Moreover, many clustering strategies have been proposed to improve the culling efficiency. Most of these techniques are used as preprocessing steps [9, 10]. Wong et al. [11] presented a continuous SCD algorithm for skeletal models and extended it to check for collisions between a deformable surface and a solid model [12]. However, these techniques have several shortcomings during animation, and their cost reduction for CD is limited. A modified framework was proposed in [13] to improve the culling efficiency of these methods. He et al. [14] recently presented a fast decomposition algorithm in which the mesh boundary is represented using hierarchical clusters and only inter-cluster collision checks are necessary; this algorithm achieves a small speedup over previous CCD algorithms.

Low-level culling: Many techniques have been proposed to reduce the number of elementary tests between triangle pairs for CCD. Govindaraju et al. and Wingo et al. [15, 16] eliminated redundant elementary tests for CCD. Hutter and Fuhrmann [17] used the bounding volumes of primitives to reduce false positives. Other methods, such as representative triangles [18] and orphan sets [2], have also been used to reduce the number of duplicate elementary tests. These low-level culling algorithms can be combined with our high-level culling method.

Reliable collision queries: Brochu et al. [19] used exact computations for reliable CCD, thereby ensuring no false negatives or false positives. Tang et al. [20] presented another exact algorithm based on Bernstein sign classification (BSC) that offers speedups of a factor of $10 - 20$ over [19]. Wang [21] introduced a useful approach based on the derivation of tight error bounds for floating-point computations. Wang et al. [22] derived tight error bounds on the BSC formulation [20] for elementary tests.

3. Overview

In this section, we present the problem definition and introduce the notation used throughout the rest of this paper. We also present an overview of the normal cone test algorithm proposed in [1].

3.1. Problem Definition

We assume that the scene of interest consists of one or many deformable objects. Each object is represented by a triangle mesh for simulation. Given two discrete time instances in a simulation, we assume that the vertices of the objects move at a constant velocity during the time interval between them. Our goal is to check whether any object exhibits any self-collision. Our approach can be used to perform both DCD and CCD on triangular meshes. For DCD, our method returns the number of potentially colliding triangle pairs. For CCD, our culling method computes the number of elementary collisions between vertex-face (VF) pairs and edge-edge (EE) pairs.

3.2. Notation

We use the following acronyms throughout the rest of the paper: BV, BVH, and BVTT stand for bounding volume, bounding volume hierarchy, and bounding volume test tree, respectively. We define a cone ($\vec{A}$, θ) in terms of $\vec{A}$, the axis, and θ, half of the apex angle of the cone (Figure 2-a). Unless otherwise specified, the angle of a cone refers to θ. For a BVH node N, N_l and N_r represent its left and right child nodes, respectively; N_{ll} and N_{lr} represent the left and right child nodes of N_l; and N_{rl} and N_{rr} represent the left and right child nodes of N_r.

3.3. Normal Cone Test

Several widely used SCD algorithms are based on the normal cone test algorithm proposed by Volino and Thalmann [1]. Given a continuous surface S bounded by a contour C, a sufficient set of criteria for no self-collision consists of both of the following sequential conditions:

- **Surface normal test:** There exists a vector $\vec{V}$ for which $(\vec{N} \cdot \vec{V}) > 0$ at every point on S, where $\vec{N}$ is the normal vector at each point on the surface.
- **Contour test:** The projection of C along the vector $\vec{V}$ does not have any self-intersections on a plane orthogonal to $\vec{V}$.

Provot [4] presented an efficient method for evaluating whether the first condition is satisfied based on normal cones, which can be computed by combining the normal vectors of individual triangles in a triangular mesh. However, the contour test has a worst-case time complexity of $\Theta(N^2)$, where N is the number of edges on the projected plane. To improve the efficiency of the normal cone test, Heo et al. [3] proposed a dual-cone culling method based on surface normal cones (SNCs) and binormal cones (BNCs). However, this method may result in false negatives in practice when it is combined with a BVH-based CD method.

4. Dual-Cone Culling Method

In this section, we briefly review the previously proposed dual-cone culling method [3] and highlight several cases in which this method may result in false negatives.

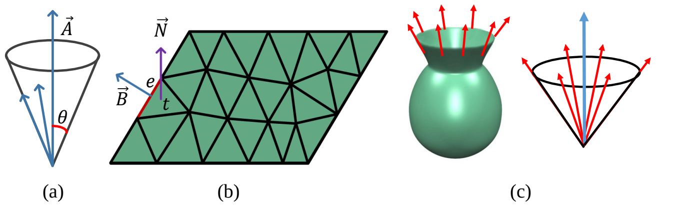


Figure 2: **Binormal Vectors and BNCs.** (a) The definition of a cone. (b) The binormal vector $\vec{B}$ computed from the boundary edge e on triangle t. (c) An example of a BNC computed from a mesh. The BNC contains all the red vectors.

4.1. Binormal Cones

Binormal vector: (Figure 2-b) The binormal vector $\vec{B}$ of an edge e of a triangle t is the cross product between the surface normal $\vec{N}$ of t and the boundary edge e.

Binormal cone (BNC): (Figure 2-c) The BNC of a mesh encompasses the binormal vectors of all boundary edges.

4.2. BVH-based Dual-Cone Culling

Given a surface S, the dual-cone method proposed in [3] uses two cones to check whether it exhibits self-collisions. The SNC $(\vec{A_n}, \theta_n)$ bounds all the surface normal vectors of S. The BNC $(\vec{A_b}, \theta_b)$ bounds all the binormal vectors of the boundary of the surface S. According to the Dual-Cone Theorem proposed in [3], if $\theta_n < \frac{\pi}{2}$ and $|\vec{A_n} \cdot \vec{A_b}| < \cos\theta_b$, then the surface exhibits no self-collision. The second condition plays the role of the contour test in the normal cone test algorithm (Section 3.3).

However, the dual-cone method presented above is very conservative for a connected surface. In many cases, due to the large angles of the BNCs generated from complete boundaries, the method may not cull meshes even when they exhibit no self-collision. Therefore, to achieve a high culling ratio, the authors combined the dual-cone method with a BVH-based CD method. During the BVH updating process, the SNC and BNC are computed for the sub-surface contained in each BV. In this method, the SNCs and BNCs of the BVH nodes are computed in a bottom-up manner. The SNC of each leaf node in a BVH can be easily computed. The BNC of each leaf node contains only the binormal vectors of the original boundary edges of the mesh, making it reasonably small. For example, in Figure 3, only the binormal vectors of the green boundary edges are bounded by BNCs. Once these cones have been computed, the two cones of each internal node can be computed by merging the cones of its two child BVH nodes. At run time, the Dual-Cone Theorem is applied to each BVH node in a top-down manner to cull the sub-meshes contained in this node that satisfy neither condition in the theorem and have no self-collision. Unless otherwise stated, in the following sections, the dual-cone method refers to the dual-cone method applied in combination with the BVH-based CD method. Although the Dual-Cone Theorem is accurate in theory, the dual-cone method yields false negatives as a result of ignoring the binormal vectors of the shared edges between adjacent sub-meshes.

4.3. False Negatives in the BVH-based Dual-Cone Method

The BVH-based dual-cone method is an approximate approach that may miss some collisions. In Figure 3, if a horizontal plane passing through the red edges (as shown in Figure 3-a) is used to partition the penetrating pipe, the dual-cone method cannot detect all self-collisions when checking the lower BV (depicted in Figure 3-b). This inaccurate culling is caused by the fact that some internal boundary edges of the object are ignored. Many internal boundary edges (shown as red curves in Figure 3) are incident on two triangles that are partitioned into two different BVs. This method considers only the binormal vectors for the original boundary edges (shown in green) and ignores the virtual boundary edges (shown in red). Therefore, many collisions that are not on the original boundary can be missed by this method if the angles of the SNCs of the sub-meshes containing these collisions are less than $\frac{\pi}{2}$.

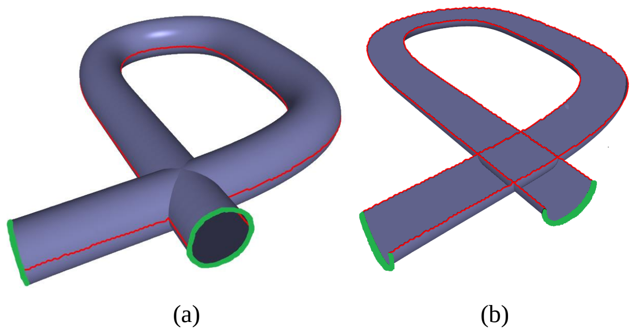


Figure 3: **Penetrating Pipe.** The edges shown in red are internal boundaries resulting from the BV splitting indicated in (a). (b) The half pipe contained in the lower BV of (a). The green edges in (b) are the original boundaries of the pipe.

The dual-cone method may also result in false negatives in cloth simulations. As shown in Figure 4, the red colliding triangle pairs are missed by the dual-cone method. These self-colliding triangle pairs always appear at locations with slight wrinkles. In the dual-cone method, BVH nodes that contain only internal triangles have no BNCs. If the angles of the SNCs of the sub-meshes contained in these nodes are less than $\frac{\pi}{2}$, then these nodes may be culled by this method even though these sub-meshes exhibit self-collisions. Many scenarios similar to the cases depicted in Figure 4 arise in cloth simulations; consequently, the dual-cone method frequently produces false negatives.

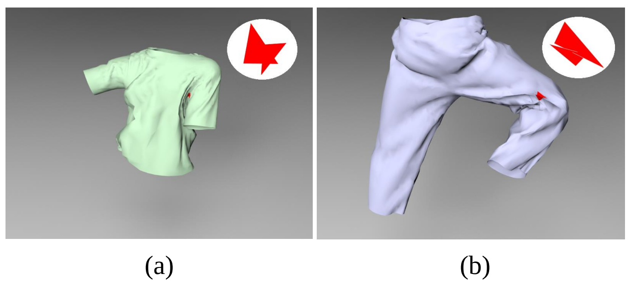
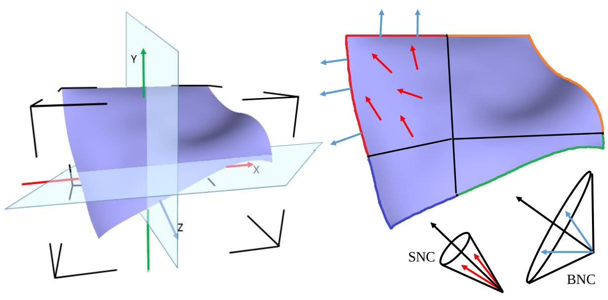


Figure 4: **Cloth.** For these two cloth simulation benchmarks, the dual-cone method cannot detect the colliding triangle pairs shown in red.



(a) (b)

Figure 5: **Surface.** For an input triangular surface (a), an AABB bounding box (shown in black) is built. At the center of this AABB box, we define two partition planes (shown in sky blue) parallel to the x and y axes, which split the black box into four sub-boxes. (b) The four resulting subsets of the boundary edges, marked in different colors. For the sub-surface in the upper left corner, we bound its surface normal vectors and the binormal vectors corresponding to its boundary edges (marked in red) into an SNC and a BNC, respectively, as shown in the lower right corner of (b). The SNCs and BNCs of the other sub-surfaces are computed in the same way.

Algorithm 1: EnhancedDualConeTest(S): Perform a self-collision test on a surface.

Input: The SNC $(\vec{A_n}, \theta_n)$ of a surface S, which is split into four sub-surfaces S_1, S_2, S_3 and S_4 with the four corresponding boundary edge subsets C_1, C_2, C_3 and C_4.
Output: True if no self-intersection on this surface, false otherwise.

```
if θn < π/2 then
    SS = {S1,S2,S3,S4};
    CC = {C1,C2,C3,C4};
    for i = 0; i ≤ 4; i++ do
        if DualConeTest(SS[i], CC[i]) == false then
            return false;
    for i = 0; i ≤ 4; i++ do
        for j = i+1; j ≤ 4; j++ do
            if ContourOverlapTest(CC[i],CC[j])==false then
                return false;
    return true;
return false;
```

Algorithm 2: DualConeTest(S_p, C_p): Perform the test for the second condition on a sub-surface.

Input: The SNC $(\vec{A}_{pn}, \theta_{pn})$ of S_p and the BNC $(\vec{A}_{pb}, \theta_{pb})$ of C_p.
Output: True if no self-intersection on this sub-boundary, false otherwise.

```
if |A⃗pn · A⃗pb| < cos θpb then
    return true;
return false;
```

5. Enhanced Dual-Cone Culling Method

To address the problems with the previously proposed BVH-based dual-cone method, we propose an enhanced BVH-based dual-cone method that can completely avoid false negatives. In section 5.1, we introduce new general culling conditions for 3D surfaces that improve upon the Dual-Cone Theorem used in the previous method. Moreover, a BVH-based culling method for use in combination with the culling conditions and a novel BVTT traversal scheme are proposed in section 5.2.

5.1. Dual-Cone-based Culling Conditions

In the normal cone test [4], given a surface, if it is sufficiently flat and the projection of its boundary edges has no intersection, then the surface is self-collision-free. A simple edge-edge test is the common method of performing the **contour test**. Similarly to the normal cone test, the Dual-Cone Theorem [3] also checks whether the surface has a sufficiently "low curvature" and uses two cones (SNC and BNC) to perform the **contour test**. However, in the Dual-Cone Theorem, the BNC of a surface bounds all binormal vectors from the entire boundary of that surface; consequently, in many cases, it is so large that it is no longer effective.

To generate smaller and more useful BNCs, we propose a novel approach in which the original boundary of a surface is split into four smaller sub-boundaries by partitioning the surface into four sub-surfaces. The binormal vectors from each sub-boundary can be bounded into a smaller BNC. Therefore, if the surface is sufficiently flat and the projections of its four sub-boundaries exhibit no self-collision and inter-collision, then the entire surface is intersection-free. These smaller BNCs can be used in self-collision detection for the four sub-boundaries in the same way that the BNC is used in the Dual-Cone Theorem [3].

Given a surface S, we partition it into four sub-surfaces, as shown in Figure 5-a. We simply partition the triangles into different sets; we do not change the topology of the surface. We first build an AABB bounding box for S. At the center of this AABB box, we define two partition planes parallel to two arbitrary axes in Cartesian coordinates. Thus, we split the bounding box into four sub-boxes. We thus partition all the triangles into four subsets based on the sub-box in which the centroid of each triangle is contained. The triangles in each subset constitute a sub-surface of the entire surface. As shown in Figure 5-b, with this splitting of the surface, the edges on the boundary contour of S are similarly partitioned into four corresponding subsets, as indicated by the four different colors.

Note that the new black internal boundary edges generated by splitting this surface are excluded from the four subsets.

For the entire surface, we bound the surface normal vectors of all triangles with a single SNC, which is used to check whether the surface has a sufficiently "low curvature". Then, for each sub-surface S_p, a corresponding SNC $(\vec{A}_{pn}, \theta_{pn})$ is defined that bounds the surface normal vectors of that sub-surface, and similarly, a BNC $(\vec{A}_{pb}, \theta_{pb})$ is defined to bound the binormal vectors associated with the corresponding boundary edge subset C_p. These two cones are used in the test for our second culling condition to check whether each sub-surface exhibits self-collision. In addition, simple EE intersection tests are ap-

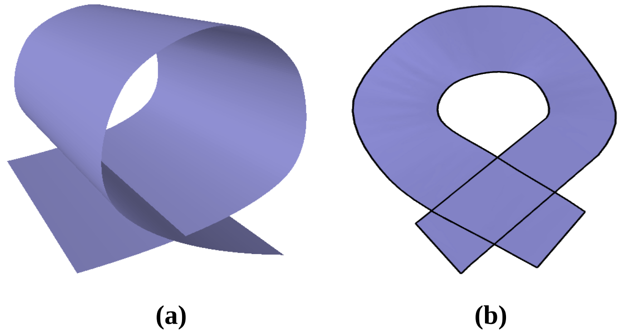


Figure 6: **Causes of Self-Collision.** Self-collisions occurring because of curvature (a) or contour shape (b).

plied to check whether there are inter-collisions among the four sub-surfaces, which is an essential step of our culling method.

Given a continuous surface S that has been partitioned into four sub-surfaces, to ensure that the surface exhibits no self-collision, it is sufficient to confirm that the following three conditions are satisfied:

- The angle of the SNC of S is less than $\frac{\pi}{2}$.
- For each sub-surface, $|\vec{A}_{pn} \cdot \vec{A}_{pb}|$ is less than $\cos\theta_{pb}$.
- The projections of two different pieces of the boundary of S along the axis of the SNC of S do not intersect on the projection plane.

The first condition is equivalent to the **surface normal test** presented in [1]. We divide the large BNC into four smaller BNCs of a more reasonable size, thereby improving the applicability of the Dual-Cone Theorem of [3]. In essence, the second and third conditions play the role of the **contour test**, drawing support from the idea of the Dual-Cone Theorem.

The axis of the SNC of the entire surface and a single point on this surface can be used to construct a projection plane for projecting the edges onto a single plane. We justify the correctness of our culling condition tests in section 5.1.1.

The pseudo-code for our culling method based on these tests is presented in Algorithm 1. **DualConeTest** returns true if the surface satisfies the second condition, and **ContourOverlapTest** returns true if the projections of the contours of a pair of sub-surfaces do not intersect. Algorithm 2 presents a more detailed explanation of **DualConeTest**. We use the BVs of the projected edges for the overlap tests and perform EE tests only for edge pairs whose BVs overlap according to **ContourOverlapTest**.

5.1.1. Explanation of Correctness

It is evident that a regular and smooth surface exhibits few self-intersections, except in the following two cases [1]:

- The surface has a sufficiently high curvature that it forms a loop and intersects with another part of itself (Figure 6-a).
- The contour of the surface has a folded shape that results in self-collisions (Figure 6-b).

In the **normal cone test**, the **surface normal test** can detect the first case, and the **contour test** can find the second one. Therefore, the **normal cone test** [1] is sufficient to determine that a surface exhibits no self-collision.

Our culling criteria are collectively equivalent to the **normal cone test**. The first condition ($\theta_n < \frac{\pi}{2}$) is equivalent to the **surface normal test**. The second and third conditions are used to guarantee no self-intersection of the projected contour; i.e., they play the role of the **contour test**. To test the second condition, the BNC of a planar curve is calculated. If the angle of the BNC is less than $\frac{\pi}{2}$, then the planar curve must have no self-intersection (as stated by the Turning Tangent Theorem [23]). When the third condition is satisfied, there is no intersection among any of the projected curve segments. Thus, by combining the second and third conditions, we obtain a conservative contour test. In summary, our culling criteria are sufficient to guarantee no self-collision among the input meshes.

5.2. Enhanced Culling Method with BVHs

Our culling criteria can be easily combined with BVHs to improve the efficiency of SCD. By virtue of the properties of the culling criteria and BVHs, our BVH-based culling method can overcome the problems with the previous BVH-based dual-cone method and ensure accurate CD. The test presented in section 5.1 can be used to check whether the surface in an intermediate BVH node exhibits self-collision. The surface contained in each intermediate node is used as the input for our culling condition test introduced in section 5.1. For the BVH node N in Figure 8, the corresponding surface is partitioned into four sub-surfaces, represented by its four grandchild nodes. Simultaneously, the boundary of this surface is also partitioned into four subsets corresponding to the four grandchild nodes. In accordance with our culling criteria, we can use the SNC of the surface in N to check whether it has a sufficiently "low curvature". Four pairs of cones and boundary edge subsets are then used to detect intra-collisions and inter-collisions among the four sub-surfaces. For each grandchild node, the associated pair of cones consists of the SNC for the corresponding sub-surface and the BNC for the corresponding boundary edge subset. In addition, we also detect inter-collisions among the four boundary edge subsets. Therefore, collisions occurring in an intermediate node can be found using criteria equivalent to those of the normal cone test. Furthermore, we can logically extend this idea to all intermediate BVH nodes that have grandchild nodes. Thus, we can apply our culling criteria in combination with BVHs in a top-down manner to perform high-level culling.

5.2.1. Internal Boundary Edges

For an object with a BVH, the internal boundary edges in the BVH nodes are incident on pairs of triangles that are partitioned into two different BVs.

Consider the example surface in Figure 7, for which a BVH is built. Take as an example the child node of the root node that is depicted in the lower branch in the figure; for this child node,

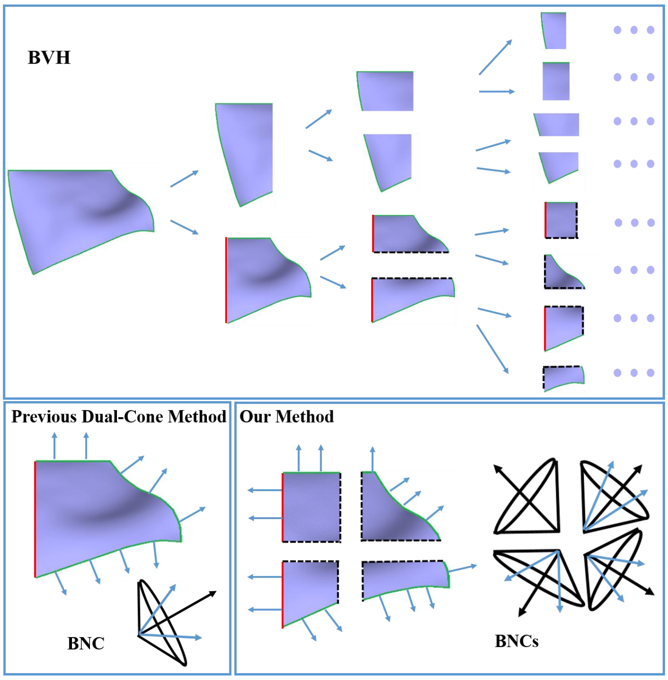


Figure 7: **Internal Boundary Edges.** For an input triangular surface, a BVH is built. For the example of the child node of the root node that is depicted in the lower branch in this figure, the previous dual-cone method consider a BNC that bounds only the binormal vectors of the original boundary edges (shown in green). By constrast, by virtue of our smaller BNCs, our method can also consider the internal boundary edges that are generated by splitting the parent surface (shown in red). The internal boundary edges that are generated by further splitting this surface into grandchild nodes are shown in black; these black edges are ignored even in our smaller BNCs. In other words, only the edges on the boundary contour of the surface are included in our BNCs. The new internal boundary edges on the contours of the four sub-surfaces are excluded, following the same principle as our culling conditions.

the BNC that bounds the binormal vectors of all boundary edges is so large that it is ineffective for high-level culling. Therefore, in the previous BVH-based dual-cone culling method, the binormal vectors on the red internal boundary edges are excluded from the BNC to make it smaller, which can result in false negatives. To address this problem, the authors of [3] proposed an extension of the method that includes internal boundary edges. In this modified method, separate dual-cones are built for these internal boundaries. However, for many surfaces in intermediate BVH nodes, if the majority of the boundary edges are internal boundary edges, the new binormal cones for the internal boundary edges may again be so large that they offer no culling effect for these surfaces. Although this method results in no false negative, maintaining such internal boundary edges can reduce its performance. In contrast to these two dual-cone methods, our method not only considers all boundary edges but also uses more useful and efficient dual-cones. In accordance with the culling conditions presented in section 5.1, the four surfaces in the grandchild nodes can be used to define four smaller BNCs that bound the binormal vectors from the corresponding boundary edge subsets. Our BVH-based method has no need to exclude the red internal boundary edges in Figure 7 because our four BNCs, which collectively bound the binor-

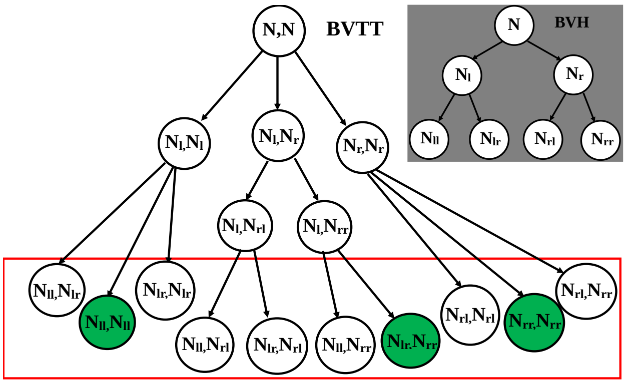


Figure 8: **BVHs and BVTTs**. For the BVH in the upper right corner, we construct a BVTT, as shown here. Only the nodes shown in green cannot pass the culling condition tests; therefore, in accordance with our traversal scheme, we continue to perform collision tests only for these three green nodes, effectively eliminating the CD tests for the other seven nodes (in the red box).

mal vectors of all boundary edges, are sufficiently small individually and thus are more useful than the BNC used in the previous BVH-based dual-cone culling method. The black internal boundary edges, which do not lie on the boundary contour of the entire surface, are excluded in our method; however, these edges will later be treated as the boundary edges of the grandchild nodes while traversing the BVH. During the traversal of the grandchild nodes, the sub-surfaces contained in these nodes are the input surfaces for our culling conditions, and the black edges now play the role of red edges and thus can also be considered in our method. The constructed BNCs are used in the test for our second culling condition, and we also test for intersections among these boundary edges. In this way, our BVH-based culling method performs exact **contour test** that fully consider the internal boundary edges.

5.2.2. Preprocessing

According to our culling conditions, we should use four subsets of the boundary edges to compute the BNCs for the surface in each intermediate BVH node. To this end, four edge sets should be collected for each intermediate node during preprocessing. Each edge set contains a subset of the boundary edges in this node. These subsets can be computed by finding the edges on the boundaries that correspond to both the node itself and its four grandchild nodes. During preprocessing, the BVHs are traversed in a bottom-up manner to collect four edge sets for each intermediate node.

5.2.3. Updating

For each frame, the BVHs, SNCs, and BNCs are updated through refitting in a bottom-up manner. The SNC of each node is computed by merging the two corresponding cones of its two child BVH nodes. For the BNCs, we compute the binormal vectors of the edges in each of the edge sets computed during preprocessing. The binormal vectors from one edge set are bounded by one BNC. Thus, the two types of cones for the BVH nodes can be computed by traversing the BVH. In this way, each intermediate BVH node, which has four grandchild

Algorithm 3: SelfCollide(N): Perform high-level culling using our criteria and the new BVH traversal scheme.

Input: A BVH node N, where S is the surface in this node, and a set of Boolean values R.
Output: No return value.

if ***IsLeaf***(N) **then**
    return;   // Traversal terminated.
if ***!IsLeaf*** *(N→LeftChild) and* ***!IsLeaf*** *c(N→ RightChild)* **then**
    // Initial value of each element in R is false.
    $R = \{R_{ll}, R_{lr}, R_{rl}, R_{rr}, R_{ll|lr}, R_{ll|rl}, R_{ll|rr}, R_{lr|rl}, R_{lr|rr}, R_{rl|rr}\}$ = {false,...,false};
    // R is updated by the following function
    if ***EnhancedDualConeTest***(S, R) **then**
        return;   // S has no self-collision.
    else
        //Check the descendants in accordance with the values in R.
        $O = \{N_{ll}, N_{lr}, N_{rl}, N_{rr}\}$;
        for $i = 0; i \leq 4; i++$ **do**
            if $R[i] == false$ **then**
                SelfCollide(O[i]);
        for $i = 0; i \leq 4; i++$ **do**
            for $j = i + 1; j \leq 4; j++$ **do**
                if $R[i + j + 4] == false$ **then**
                    Collide(O[i],O[j]);
else
    SelfCollide(N→LeftChild);
    SelfCollide(N→ RightChild);
    Collide(N→LeftChild,N→RightChild);

nodes, is associated with one SNC and four BNCs. Each of the BNCs corresponds to one of the grandchild nodes because the four grandchild nodes partition the boundary into four parts by splitting the entire surface into four sub-surfaces. Then, the SNC of each grandchild node and the corresponding BNC are used in the test for the second condition as described in section 5.1.

5.2.4. Run time

During run time, our self-collision check starts at the root node of the BVH and traverses the BVH in a top-down manner. For a scene with deformable objects with the BVH shown in the upper corner of Figure 8, the execution of the self-collision detection algorithm corresponds to the traversal of its BVTT, as shown in Figure 8. A node (A, B) in the BVTT represents the collision check between nodes A and B of the given BVH. When applying our culling conditions for these BVTT nodes (N, N), which corresponds to checking for self-collisions among all the nodes below the intermediate node N of the BVH, this intermediate node and its four grandchild nodes are considered to check whether that node can be culled. For the example of BVH node N in Figure 8, four of the BVTT nodes for SCD (in the red box) correspond to four tests for the second condition, and the other nodes in the box correspond to six tests

Algorithm 4: EnhancedDualConeTest(S, R): Perform a self-collision test on the surface in one BVH node.

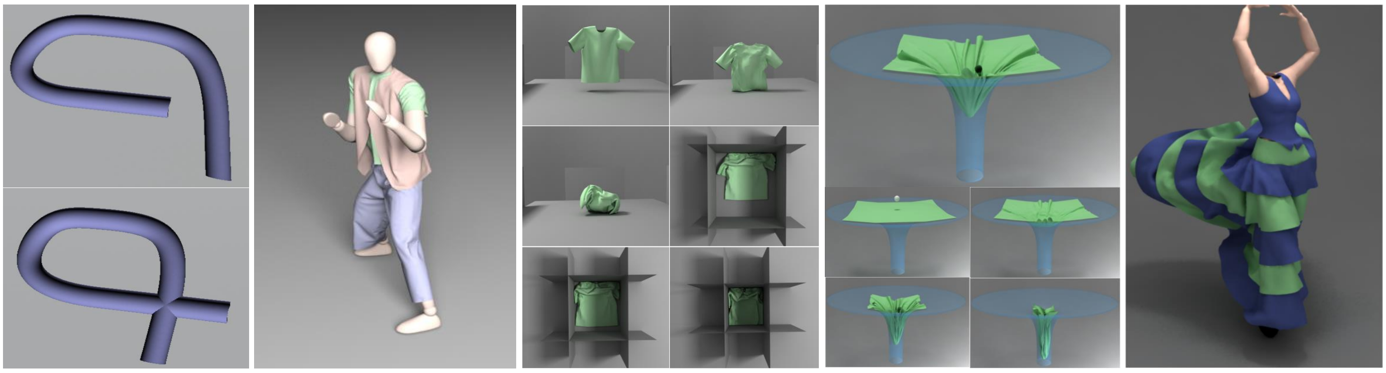
Input: A surface S, which has the same configuration as the surface in Algorithm 1, and a set of Boolean values R.
Output: True if no self-intersection on the surface, false otherwise.

if $\theta_n < \frac{\pi}{2}$ **then**
    $SS = \{S_1, S_2, S_3, S_4\}$;
    $CC = \{C_1, C_2, C_3, C_4\}$;
    for $i = 0; i \leq 4; i++$ **do**
        $R[i]$=**DualConeTest**($SS[i]$);
    for $i = 0; i \leq 4; i++$ **do**
        for $j = i + 1; j \leq 4; j++$ **do**
            t=$i + j + 4$;
            $R[t]$= **ContourOverlapTest**($CC[i]$,$CC[j]$);
    if *all elements in R are true* **then**
        return true;
return false;

for the third condition. In accordance with our culling conditions, in the worst case, we will perform all ten tests enclosed in the red box in the BVTT of N in Figure 8. The traditional BVTT traversal scheme is a simple top-down traversal scheme. When a BVTT node cannot be culled, SCD and inter-collision detection will then be performed at the subsequent level in the hierarchy.

However, based on the properties of our criteria, we propose a more efficient traversal scheme as follows: During our culling tests on N, the results of the ten tests in the red box are recorded. If the surfaces contained in N and the grandchild nodes of N all satisfy our self-collision culling criteria, then this surface in N is collision-free, and the following BVTT nodes need not be traversed. Otherwise, in accordance with the previously recorded results, our culling method continues to be performed on only the BVTT nodes that correspond to grandchild nodes and cannot pass the relevant tests. In this way, we can eliminate many redundant tests. When each of the grandchild nodes is traversed, a new self-collision test of a surface is initiated, and the surface in that node is treated as the input for testing our culling criteria presented in section 5.1.

Based on Algorithm 1, the overall algorithm for our entire culling method is shown in Algorithm 3. Given a deformable object with a BVH, the process of checking for self-collisions begins at the root node of the BVH and traverses it in a top-down manner. For a BVH node N with grandchild nodes, we perform our culling method using its four grandchild nodes N_{ll}, N_{lr}, N_{rl} and N_{rr}. R is a set of Boolean values that records the results of the ten tests enclosed in the red box in Figure 8. In addition, in Algorithm 4, we describe how the new **EnhancedDualConeTest** function updates R. This function can be regarded as a variant of the function described in Algorithm 1. The input surface S again is split into four sub-surfaces and four boundary edge subsets corresponding to its grandchild nodes, which are used in Algorithm 4 in the same manner as in Algorithm



(a) Pipe **(b) Karate** **(c) T-shirt** **(d) Funnel** **(e) Flamenco**

Figure 9: **Benchmarks.** We use five challenging benchmarks involving deformable models and cloth simulations for performance comparisons between our DCD and CCD algorithms and previous methods.

Bench-marks	DCD					CCD									
	# of Triangle-Triangle Intersections					# of VF Pairs					# of EE Pairs				
	Dual-Cone Method		Our Method	AABB Only	NCT	Dual-Cone Method		Our Method	AABB Only	CBC	Dual-Cone Method		Our Method	AABB Only	CBC
Pipe	1433	522	1955	1955	1955	7613	184	7797	7797	7797	34287	1128	35415	35415	35415
Karate	157526	68	157594	157594	157594	15916	10	15926	15926	15926	75436	19	75455	75455	75455
T-shirt	33899	252	34151	34151	34151	877	0	877	877	877	10436	17	10453	10453	10453
Funnel	7378	0	7378	7378	7378	1809	0	1809	1809	1809	6523	0	6523	6523	6523
Flamenco	40459	0	40459	40459	40459	8206	0	8206	8206	8206	22667	0	22667	22667	22667

# of false negatives

Figure 10: **Number of Collision Queries.** We compare the numbers of collision queries performed in our enhanced dual-cone method, in the dual-cone method without internal edges [3], and in other previous methods. As this figure shows, our method results in exactly the same numbers of collisions as those of the other three culling methods; however, the dual-cone method can generate false negatives for Pipe, T-shirt and Karate (with correspondingly fewer collisions).

1. After all ten tests for our second and third conditions have been performed, the first four values in R correspond to the SCD results for the sub-surfaces contained in the grandchild nodes, and the six remaining values represent whether each pair of sub-surfaces exhibits collision. If all these tests are satisfied, then there is no need to traverse the grandchild nodes to check for collisions. **Collide** generates a list of the leaf nodes that will need to be traversed in the next round of CD processing.

Moreover, we can extend our culling method to CCD by adopting the same approach used in [3] to compute the SNCs and BNCs. The continuous contour test (CCT) method proposed in [2] is used to check whether contour edge sets overlap. Compared with the previous dual-cone method, our method is slightly slower because it requires more tests to be performed; however, it also results in no false negatives.

6. Implementation and Results

In this section, we describe our implementation and demonstrate the accuracy of our algorithm.

6.1. Implementation

We implemented our algorithms on a standard PC (Intel i7-4790K CPU @4.00 GHz, 32 GB of RAM, 64-bit Windows 7 OS) in C++. We also implemented the dual-cone method [3] on the same CPU (also in C++). We performed a high-level culling procedure including both our culling method and low-level culling techniques that can eliminate duplicate elementary tests [2]. After these culling computations, we performed triangle-triangle intersection tests for DCD and exact elementary tests for CCD [20]. We present performance comparisons of our algorithm with the following previous methods:

- **Dual-Cone Method:** This is the algorithm without internal edges that we describe in Section 4, which misses collisions on many benchmarks. This method has been implemented for DCD and CCD, in combination with AABB hierarchies and BSC elementary tests [20]. As our experiments show, our method can make up for the defects of this method.

- **NCT:** This method corresponds to the implementation of the normal cone test of [4] for DCD. It is based on the **surface normal test** and the **contour test**.

- **CBC:** This is the continuous normal cone algorithm for CCD [2] in combination with AABB culling and BSC elementary tests. This method extends the normal cone test to tests for CCD, namely, the continuous normal cone test and the continuous contour test. In this method, the contour test is transformed into a test for intersection between two edges that lie on the same plane.

- **AABB only:** In this method, no self-collision culling is performed; only low-level culling algorithms are used to eliminate duplicate elementary tests. The AABB approach is used to determine the BVs, and reliable elementary tests are performed using BSC [20].

- **Dual-Cone Method (Internal):** This algorithm is the previous BVH-based dual-cone method modified to also consider internal edges, as proposed in [3] to address the problem of false negatives. Separate BNCs are computed for the internal boundary edges.

6.2. *Benchmarks*

We used five benchmarks related to different simulation scenarios for our performance evaluations:

- **Pipe:** (Figure 9-a) A hollow pipe with 78K triangles lies on the ground, and one end of the pipe can intersect with the other. This benchmark has a high number of self-collisions.

- **Karate:** (Figure 9-b) A boy wearing three pieces of cloth (with 127K triangles) is practicing karate. We count only the number of self-collisions for each piece of cloth.

- **T-shirt:** (Figure 9-c) A T-shirt (with 10K triangles) is stuffed into a small box, which generates numerous self-collisions of the cloth.

- **Funnel:** (Figure 9-d) A piece of cloth with 64K triangles falls into a funnel and folds to fit into the funnel, exhibiting many self-collisions.

- **Flamenco:** (Figure 9-e) A flamenco dancer performs while wearing a dress (with 49K triangles) with ruffles, which has numerous self-intersections.

The inputs for the Flamenco and Pipe benchmarks are given as discrete keyframes. Karate, T-shirt and Funnel were generated using a cloth simulation system. We used the linearly interpolated motion of the vertices between keyframes to check for inter-object collisions and self-collisions.

We integrated our CD algorithm into a cloth simulation system, which was then used to generate the entire simulation for each of the Karate, T-shirt and Funnel benchmarks. This simulator performs the implicit integration described in [24] and uses the repulsion forces presented in [25] along with CCD computations to avoid interpenetration.

Figure 10 shows the numbers of triangle-triangle intersections for DCD and the numbers of exactly colliding elementary pairs (VF and EE pairs) for CCD found throughout the entire CD process using our high-level culling algorithm and the other four methods on these benchmarks. It also reports the numbers of false negatives generated by the previous dual-cone method. Compared with the AABB-hierarchy-based culling method with BSC [20], the previous dual-cone method results in false negatives on Pipe, Karate and T-shirt for both CCD and DCD, whereas our method does not miss any collisions on these benchmarks. On the other two benchmarks, our method and the previous dual-cone method yield the same results. We illustrate the speedups of our algorithm in comparison with the other three high-level culling methods for each benchmark in Figure 11. The speedups of our method range from minor to significant. We also compare the accuracy and time consumption of our method, the dual-cone method and the dual-cone method (internal) for the above benchmarks. We observe that the dual-cone method (internal) produces no false negatives, as reported in [3], but it is slower than the other two methods. The average times (in ms) required for DCD and CCD queries in these three methods are presented in Figure 12.

Benchmarks	DCD		CCD	
	NCT	AABB only	CBC	AABB only
Pipe	1.05X	1.08X	1.04X	1.06X
Karate	1.14X	1.52X	1.06X	1.16X
T-shirt	1.08X	1.15X	1.23X	1.27X
Funnel	1.29X	1.77X	1.21X	1.38X
Flamenco	1.45X	1.82X	1.14X	0.92X

Figure 11: **Performance and Comparison.** We present the speedups of our algorithm in comparison with the NCT [4], CBC [2] and AABB-hierarchy-based culling methods for each benchmark.

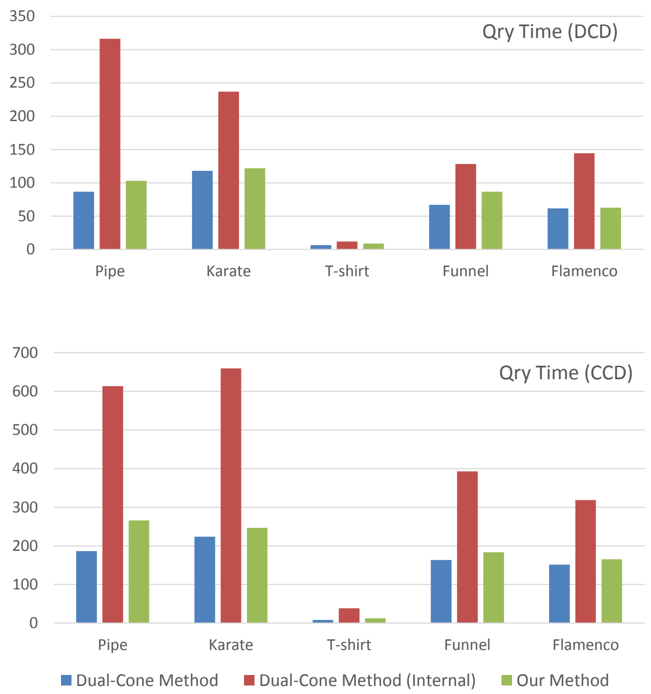


Figure 12: **Time Consumption Comparison.** We compare the average times (in ms) required for DCD and CCD queries in our method, the dual-cone culling method [3] and the dual-cone method (internal) for each benchmark.

We also compare the numbers of self-collision tests and inter-collision tests required for CCD based on our culling criteria during BVTT traversal using our new scheme and the simple top-down traversal scheme. In this evaluation, we used the depth-first traversal algorithm to traverse the BVTT in the simple traversal scheme. Figure 13 shows the average numbers of additional self-collision tests and inter-collision tests required in the simple top-down traversal scheme compared with our traversal scheme. The corresponding minor speedups of the proposed scheme over the simple scheme are also given in this figure. Moreover, we also tested the previous BVH-based dual-cone method with our BVTT traversal scheme. Compared with the simple top-down traversal scheme, our scheme offers no advantage in this case because it traverses BVTT layers more deeply than the simple scheme does, which reduces the performance. Moreover, our culling conditions force the emergence of our new traversal scheme, which is more suitable for our culling method.

Benchmarks	Pipe	Karate	T-shirt	Funnel	Flamenco
# of Self-Collision Tests	19897	7351	1636	2674	4270
# of Inter-Collision Tests	252204	9727	12062	35788	7826
Speedup	1.13X	1.05X	1.06X	1.08X	1.03X

Figure 13: **Comparison of two BVTT traversal schemes.** We compare the average numbers of self-collision tests and inter-collision tests required when using the simple top-down traversal scheme and our traversal scheme. The average numbers of additional tests required when using the simple scheme are shown in this figure. Because of these differences, our scheme is slightly faster than the simple scheme.

6.3. Analysis

Our enhanced dual-cone method produces no false negative for either CCD or DCD, both in theory and in practice (see Figure 10). This is because our method not only computes the binormal vectors of the original boundary edges of objects but also considers internal edges. In the Pipe, Karate and T-shirt benchmarks, there are many collisions on internal triangles, and because the previous dual-cone method does not compute binormal vectors for the internal boundary edges of the sub-meshes that exhibit these collisions, these collisions may be missed. By contrast, for Funnel and Flamenco, the dual-cone method does not miss any collisions because for these two benchmarks, simply checking whether the angles of the SNCs are less than $\frac{\pi}{2}$ is sufficient to find all collisions. In fact, because the second condition in the dual-cone method ignores the internal edges, it cannot completely replace the contour test that is performed as part of the normal cone test presented in [1]. Meanwhile, although the new BVTT traversal scheme proposed for use in our method cannot accelerate the BVH updating process, it can eliminate many redundant tests in the BVTT and thus reduce the time spent traversing the BVTT.

We also compared our method with the other techniques in terms of time consumption. Compared with the NCT, CBC and AABB-hierarchy-based culling methods, our method requires less time for high-level culling because it performs fewer EE intersection tests and thus yields more efficient culling results (see Figure 11). However, compared with the previous dual-cone method, our method requires the computation of more binormal vectors and the performance of more EE tests and thus is slightly slower, as shown in Figure 12. As reported in [3], although the dual-cone method (internal) prevents the occurrence of false negatives, the consideration of the additional cones can significantly degrade its time performance. For some intermediate BVH nodes whose boundary edge sets contain few or no original boundary edges, the angles of the BNCs for the additional internal boundary edges are no less than $\frac{\pi}{2}$, so these BNCs cannot play an effective role in dual-cone culling. Although they consider the internal boundary edges, their inclusion prevents the traversal of the BVTT from stopping as soon as possible, which slows the performance of this method. Therefore, the essential concept of our method is to consider BNCs for all boundary edges in the BVH nodes while simultaneously making these BNCs more useful.

7. Conclusion, Limitations and Future Work

Inspired by the previously proposed dual-cone culling method, we present a reliable algorithm for performing self-collision culling on complex deformable models. We introduce new conditions for checking whether a surface exhibits self-collisions, a BVH-based hierarchical culling method using these dual-cone criteria, and a new hierarchical traversal scheme. Unlike the previously proposed dual-cone culling method, our method can reliably detect all self-collisions on the benchmarks used for testing, thereby overcoming the defects of the original dual-cone method.

Our approach has some limitations. First, our method uses the conditions of the Dual-Cone Theorem, which yields conservative results, to check whether a boundary exhibits intersections. Dual-cone-based self-collision culling works well only when the resulting meshes do not exhibit high variation in curvature. In addition, since our method requires more information than is required by previous methods, more computations are required for BVH updates.

There are many potential avenues for future work. We would like to parallelize our approach on multi-core CPUs and GPUs, similar to the work reported in [26] and [27]. And we prefer to combine our method with BVTT front [28, 29] and apply this technique into the self-collision culling in cloth simulation [30]. Furthermore, we would also like to optimize our method to achieve faster BVH updating. Finally, we would like to integrate our algorithm with other simulation systems, such as hair simulation systems and finite element modeling systems.

Acknowledgements

This work was supported by the National Key R&D Program of China [2017YFB1002703]; NSFC [61732015, 61572423, 61572424]; the Science and Technology Project of Zhejiang

Province [2018C01080]; and Zhejiang Provincial NSFC [LZ16F-020003]. We also thank Zhihua Liu for useful discussions.

References

[1] Volino P, Thalmann NM. Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. In: Computer Graphics Forum; vol. 13. Wiley Online Library; 1994, p. 155–66.

[2] Tang M, Curtis S, Yoon SE, Manocha D. ICCD: Interactive continuous collision detection between deformable models using connectivity-based culling. IEEE Transactions on Visualization and Computer Graphics 2009;15(4):544–57.

[3] Heo JP, Seong JK, Kim D, Otaduy MA, Hong JM, Tang M, et al. FASTCD: fracturing-aware stable collision detection. In: Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. Eurographics Association; 2010, p. 149–58.

[4] Provot X. Collision and self-collision handling in cloth model dedicated to design garments. Springer; 1997.

[5] Mezger J, Kimmerle S, Etzmuß O. Hierarchical techniques in collision detection for cloth animation 2003;.

[6] Schvartzman SC, Pérez ÁG, Otaduy MA. Star-contours for efficient hierarchical self-collision detection. In: ACM Transactions on Graphics (TOG); vol. 29. ACM; 2010,.

[7] Barbič J, James DL. Subspace self-collision culling. In: ACM Transactions on Graphics (TOG); vol. 29. ACM; 2010,.

[8] Zheng C, James DL. Energy-based self-collision culling for arbitrary mesh deformations. ACM Transactions on Graphics (TOG) 2012;31(4).

[9] Ehmann SA, Lin MC. Accurate and fast proximity queries between polyhedra using convex surface decomposition. In: Computer Graphics Forum; vol. 20. Wiley Online Library; 2001, p. 500–11.

[10] Wong SK, Baciu G. Continuous collision detection for deformable objects using permissible clusters. The Visual Computer 2015;31(4):377–89.

[11] Wong SK, Lin WC, Hung CH, Huang YJ, Lii SY. Radial view based culling for continuous self-collision detection of skeletal models. ACM Transactions on Graphics (TOG) 2013;32(4).

[12] Wong SK, Cheng YC. Continuous self-collision detection for deformable surfaces interacting with solid models. In: Computer Graphics Forum; vol. 33. Wiley Online Library; 2014, p. 143–53.

[13] Wong SK, Lin WC, Wang YS, Hung CH, Huang YJ. Dynamic radial view based culling for continuous self-collision detection. In: Proceedings of the 18th meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. ACM; 2014, p. 39–46.

[14] He L, Ortiz R, Enquobahrie A, Manocha D. Interactive continuous collision detection for topology changing models using dynamic clustering. In: Proceedings of the 19th Symposium on Interactive 3D Graphics and Games. ACM; 2015, p. 47–54.

[15] Govindaraju NK, Knott D, Jain N, Kabul I, Tamstorf R, Gayle R, et al. Interactive collision detection between deformable models using chromatic decomposition. In: ACM Transactions on Graphics (TOG); vol. 24. ACM; 2005, p. 991–9.

[16] Wong WSK, Baciu G. A randomized marking scheme for continuous collision detection in simulation of deformable surfaces. In: Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications. ACM; 2006, p. 181–8.

[17] Hutter M, Fuhrmann A. Optimized continuous collision detection for deformable triangle meshes 2007;.

[18] Curtis S, Tamstorf R, Manocha D. Fast collision detection for deformable models using representative-triangles. In: Proceedings of the 2008 symposium on Interactive 3D graphics and games. ACM; 2008, p. 61–9.

[19] Brochu T, Edwards E, Bridson R. Efficient geometrically exact continuous collision detection. ACM Transactions on Graphics (TOG) 2012;31(4).

[20] Tang M, Tong R, Wang Z, Manocha D. Fast and exact continuous collision detection with Bernstein sign classification. ACM Transactions on Graphics (TOG) 2014;33(6).

[21] Wang H. Defending continuous collision detection against errors. ACM Transactions on Graphics (TOG) 2014;33(4).

[22] Wang Z, Tang M, Tong R, Manocha D. TightCCD: Efficient and robust continuous collision detection using tight error bounds. In: Computer Graphics Forum; vol. 34. Wiley Online Library; 2015, p. 289–98.

[23] Toponogov VA. Differential geometry of curves and surfaces. Springer; 2006.

[24] Baraff D, Witkin A. Large steps in cloth simulation. In: Proceedings of the 25th annual conference on Computer graphics and interactive techniques. ACM; 1998, p. 43–54.

[25] Tang M, Manocha D, Otaduy MA, Tong R. Continuous penalty forces. ACM Trans Graph 2012;31(4):107–1.

[26] Tang M, Manocha D, Lin J, Tong R. Collision-streams: fast GPU-based collision detection for deformable models. In: Symposium on interactive 3D graphics and games. ACM; 2011, p. 63–70.

[27] Zhang X, Kim YJ. Scalable collision detection using p-partition fronts on many-core processors. IEEE Transactions on Visualization and Computer Graphics 2014;20(3):447–56.

[28] Wang T, Liu Z, Tang M, Tong R, Manocha D. Efficient and reliable self-collision culling using unprojected normal cones. Computer Graphics Forum 2017;36(8):487–98.

[29] Wang X, Tang M, Manocha D, Tong R. Efficient BVH-based collision detection scheme with ordering and restructuring. Computer Graphics Forum (Proceedings of Eurographics 2018) 2018;37(2).

[30] Tang M, Liu Z, Tong R, Manocha D. PSCC: Parallel self-collision culling with spatial hashing on GPUs. Proceedings of I3D 2018 2018;.