Simple Culling Methods for Continuous Collision Detection of Deforming Triangles

Xinyu Zhang, *Member, IEEE,* and Young J. Kim, *Member, IEEE*

Abstract—We present a simple and efficient approach for continuous collision detection of deforming triangles based on conservative advancement. The efficiency of our approach is due to a sequence of simple collision-free conditions for deforming triangles. In our experiment, we show that our CCD algorithm achieves $2 \sim 30$ times performance improvement over existing algorithms for triangle primitives.

Index Terms—Continuous Collision Detection, Conservative Advancement, Distance Computation.

1 Introduction

The goal of continuous collision detection (CCD) is to calculate the earliest time instance when two moving objects come into contact. Recently, CCD between deformable objects has drawn much attention in computer graphics and computer animation [1]. To accelerate the CCD performance, existing CCD algorithms focus on minimizing the number of elementary CCD tests using bounding volume hierarchies (BVHs) or topological information available in the models. However, these algorithms finally boil down to performing elementary CCD tests between triangle primitives. There are two known elementary CCD approaches for deforming triangles: algebraic equation solvers and conservative advancement. Compared to the BVH-level CCD acceleration techniques, elementary CCD has been relatively poorly studied and deserves more research efforts.

Conservative advancement (CA) is an efficient method to perform CCD and has been applied to convex [2], non-convex [3], articulated [4] and deformable objects [5] in the past. For a pair of deforming triangles, CA can be carried out in two manners: feature-level and triangle-level CA queries. The feature-level CA query includes six face-vertex and nine edge-edge CA queries, and the time of contact can be obtained as a minimum of the 15 CA queries. On the other hand, CA can be directly applied to a triangle pair (i.e. triangle-level CA), which is also known as local advancement [6].

Main Results: In this paper, we present a simple and efficient culling algorithm to perform an elementary CCD query for a pair of deforming triangles. Our algorithm uses feature-level queries, and reduces the number of feature-wise distance computations, the known computational bottleneck in CA-based approaches [3], by using a sequence of conservative culling tests. In our experiments, when the deforming triangles do not collide, on average, we observe more than 30 times performance improvement over a straightforward feature-level CA and 5$\sim$6 times performance improvement over a triangle-level CA based on [6]. In a more practical setting [7] where the chances of triangle collisions are 5% or lower, we observe 6$\sim$7 and 2 times performance improvement over a straightforward feature-level CA and triangle-level CA test respectively. We also measure the performance of our algorithm on the well-known UNC dynamic benchmark consisting of three highly complicated simulations and observe, on average, 1.3$\sim$8.8 times performance improvements over that of other existing algorithms, including the straightforward feature-level CA, triangle-level CA and cubic solver.

Organization: We organize the rest of this paper as follows. We present the main idea of our collision culling approach in Section 3. We derive our culling conditions for a face-vertex case in Section 4 and for an edge-edge case in Section 5. In Section 6, we present triangle-level culling tests utilizing the culling results from the face-vertex and edge-edge cases. The entire algorithm based on the culling techniques is explained in Section 7. We show our implementation results and comparative studies in Section 8. Finally, we conclude this paper and propose possible future work in Section 9.

2 Related Work

2.1 CCD for Rigid and Articulated Objects

CCD algorithms for rigid and articulated models can be classified into: algebraic equation solvers [8], [9], [10], [11], swept volume formulations [12], [13],

• *Xinyu Zhang and Young J. Kim are with the Department of Computer Engineering, Ewha Womans University, Seoul, South Korea. E-mail: {zhangxy,kimy}@ewha.ac.kr*

adaptive bisection approaches [14], [15], kinetic data structures [16], [17], [18], Minkowski sum-based approaches [19], conservative advancement [2], [3], [4], [5] and hardware-assisted approaches [20]. For more extensive survey on these algorithms, we refer readers to see [21].

2.2 CCD for Deformable Objects

2.2.1 Elementary CCD Tests

An approach based on algebraic equation solvers was first introduced by Moore and Wilhelms [22] using fifth-order algebraic equations. These equations were further reduced to cubic by considering co-planar constraints, in which detecting collisions between deforming triangles corresponds to performing pairwise six face-vertex and nine edge-edge elementary tests and each elementary test requires solving a cubic algebraic equation [23], [24]. The first work of using CA for deforming objects was reported by Tang et al. [6], which was extended from [3], [4], [5]. The CA-based algorithms are able to avoid solving high-order algebraic equations since these solvers are known relatively expensive to evaluate for time-demanding applications.

2.2.2 CCD Acceleration Techniques

In order to reduce the number of redundant elementary CCD tests, connectivity (i.e. adjacency) or coplanarity was utilized in [7], [25], [26], [27], [28], [29].

Recently, Barbic and James [30] presented a self-collision culling algorithm for reduced deformable models. Schvartzman et al. [31] presented a star-contour test as a sufficient condition to determine whether the contour of a projected surface patch is collision-free or not. These approaches are designed for to reduce the number of elementary tests. Other high-level culling techniques for collision detection include the approaches utilizing hardware-supported visibility queries [32], [33] and chromatic decomposition [28].

3 Overview

Given two successive configurations at $t = 0$ and 1 of a deforming triangle, the CCD problem typically requires a motion that continuously interpolates the two configurations. Like many other existing CCD algorithms such as [23], [6], [24], we linearly interpolate the corresponding positions at $t = 0$ and 1. In other words, any point on the given deforming triangle undergoes a motion with a constant velocity over the entire time interval.

For such a linearly interpolating motion, CCD for deforming triangles is relatively straightforward to perform using CA. For feature-level CA, elementary tests need two computational components: closest distance d between a feature pair (face-vertex or edge-edge), and their relative motion bound μ. Then, the two features can safely advance by the step size $\Delta t = \frac{d}{\mu}$ without creating a collision. This procedure is repeated until d becomes less than a user-provided threshold. Note that the distance computation between a pair of features can be rather complicated and expensive, because, for example, in the face-vertex case, the vertex could be closest to the interior, an edge or a vertex of the triangle. Moreover, in case of triangle-level CA, d and μ should be calculated for the entire triangle.

We now present some propositions on which our culling approach is based. Let us first consider the face-vertex CA test. Obviously, if a point (or a vertex) and a triangle (or a face) collide, the point collides with the plane containing the triangle. The contrapositive of this statement is true as well:

Proposition 3.1 *: (FV Non-Colliding Proposition A) If a point does not collide with the plane containing a triangle during the motion, the point and the triangle do not collide.*

If the above proposition is not satisfied, the collision test for face-vertex is inconclusive, and it is further checked if the point's projection onto the plane collides with a circle circumscribing the triangle. Thus, its contrapositive is:

Proposition 3.2 *: (FV Non-Colliding Proposition B) If a point's projection on the plane containing a triangle lies outside a bounding circle of the triangle and does not collide with it during the motion, the point and the triangle do not collide.*

We attain similar propositions for the edge-edge CA test as follows:

Proposition 3.3 *: (EE Non-Colliding Proposition A) If the lines containing two edges do not collide during the motion, the two edges do not collide.*

Proposition 3.4 *: (EE Non-Colliding Proposition B) If bounding circles of two edges do not collide during the motion on the plane spanned by the two edges, the two edges do not collide.*

As we will see in the next two sections, distance computation involved in these propositions is computationally much cheaper than that for EE or FV feature pairs. For example, Proposition 3.1 involves distance computation for a point against a plane, which requires merely one dot product operation once the normal of the plane was obtained (also see Eq. 1). We also refer readers to Section 8 for experimental validations on these observations.

Based on the aforementioned propositions, we derive our culling conditions for face-vertex CA tests in Section 4 and for edge-edge CA tests in Section 5.

4 FACE-VERTEX CULLING

For the face-vertex case, we first check if the vertex collides with the plane containing the face. Based on Proposition 3.1, we present a culling condition which, if satisfied, guarantees that the deforming face does not collide with the vertex. If the condition is still not satisfied, we present another culling condition based on Proposition 3.2. Using this condition, we further check if the vertex's projection onto the plane collides with a bounding circle of the triangle.

We first explain how to compute the two essential components needed for conservative advancement: closest distance (Section 4.1) and relative motion (Sections 4.2 and 4.3), then present our culling conditions in Section 4.4.

4.1 Closest Distance

As depicted in Fig. 1, let $\vec{x}_1\vec{x}_2\vec{x}_3$ be a deforming triangle and let $\vec{x}_4$ be a vertex. $\vec{n}_P$ denotes the normal of the plane containing the triangle $\vec{x}_1\vec{x}_2\vec{x}_3$ and $\vec{n}_{FV}$ denotes the vector that realizes the closest distance d_{FV} from the triangle $\vec{x}_1\vec{x}_2\vec{x}_3$ to the vertex $\vec{x}_4$. Let d_P be the distance between the vertex $\vec{x}_4$ and the plane containing the triangle $\vec{x}_1\vec{x}_2\vec{x}_3$ and then d_P and $\vec{n}_P$ can be calculated as

$$d_P = \vec{x}_{41} \cdot \vec{n}_P, \quad \text{where} \quad \vec{n}_P = \frac{\vec{x}_{21} \times \vec{x}_{31}}{|\vec{x}_{21} \times \vec{x}_{31}|} \tag{1}$$

where we use the shorthand $\vec{x}_{ij}$ to denote the vector $\vec{x}_i - \vec{x}_j$. Note that d_P is a signed distance and its sign indicates where $\vec{x}_4$ lies relative to the plane containing $\vec{x}_1\vec{x}_2\vec{x}_3$:

1) if $d_P > 0$, $\vec{x}_4$ lies on the positive side of the plane[1];
2) if $d_P = 0$, $\vec{x}_4$ lies on the plane;
3) if $d_P < 0$, $\vec{x}_4$ lies on the negative side of the plane.

From our experimental statistics (see Fig. 5), computing d_P is much cheaper than d_{FV}. While computing d_{FV} requires handling difference cases, computing d_P requires merely one dot product.

As depicted in Fig. 1-(left), the closest distance from the vertex $\vec{x}_4$ to the plane containing $\vec{x}_1\vec{x}_2\vec{x}_3$ is realized by $\vec{x}_4$ and its projection $\vec{x}_4^p$ onto the plane. As shown in Fig. 1-(right), we display a bounding circle of the triangle with a dotted line and we denote its center and radius by $\vec{c}$ and r, respectively. Then, the distance d_C from the center $\vec{c}$ to $\vec{x}_4^p$ is calculated as

$$d_C = |\vec{x}_4^p - \vec{c}| \quad \text{where} \quad \vec{x}_4^p = \vec{x}_4 - d_P\vec{n}_P \tag{2}$$

Note that in contrast to d_P, d_C is an unsigned distance.

1. The positive side of the plane is the half space to which the plane normal points.

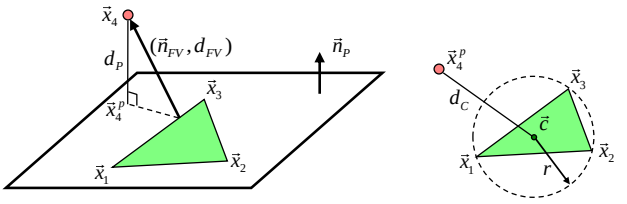


Fig. 1. Face-Vertex Case. First check if the vertex $\vec{x}_4$ collides with the plane containing the triangle $\vec{x}_1\vec{x}_2\vec{x}_3$ and then check if the vertex $\vec{x}_4^p$ collides with a bounding circle of the triangle $\vec{x}_1\vec{x}_2\vec{x}_3$.

4.2 Relative Face-Vertex Motion

We now describe how we formulate relative face-vertex motion, which will be used to evaluate the motion bound (see Section 4.3) and eventually used to deduce our culling conditions (see Section 4.4).

Let $\vec{x}$ be a point interior to the triangle $\vec{x}_1\vec{x}_2\vec{x}_3$. With barycentric coordinates ω_1, ω_2 and ω_3, $\vec{x}$ can be expressed as

$$\vec{x} = \omega_1\vec{x}_1 + \omega_2\vec{x}_2 + \omega_3\vec{x}_3 \tag{3}$$

where $\omega_1, \omega_2, \omega_3 \in [0, 1]$ and $\omega_1 + \omega_2 + \omega_3 = 1$. We assume that, for any point $\vec{x}$, its barycentric coordinates $\omega_1, \omega_2, \omega_3$ are constant during the deformable motion. Then, by taking the derivatives on the both sides, the above formula results in

$$\vec{v} = \omega_1\vec{v}_1 + \omega_2\vec{v}_2 + \omega_3\vec{v}_3 \tag{4}$$

where $\vec{v}$ is $\vec{x}$'s velocity. Thus, the relative velocity between the point $\vec{x}$ and the vertex $\vec{x}_4$ is

$$\vec{v} - \vec{v}_4 = \omega_1\vec{v}_1 + \omega_2\vec{v}_2 + \omega_3\vec{v}_3 - \vec{v}_4 \tag{5}$$

Since $\omega_1 + \omega_2 + \omega_3 = 1$, the above formula can be represented as

$$\begin{aligned} \vec{v} - \vec{v}_4 &= (1 - \omega_2 - \omega_3)\vec{v}_1 + \omega_2\vec{v}_2 + \omega_3\vec{v}_3 - \vec{v}_4 \\ &= \omega_2(\vec{v}_2 - \vec{v}_1) + \omega_3(\vec{v}_3 - \vec{v}_1) + \vec{v}_1 - \vec{v}_4 \\ &= \omega_2\vec{v}_{21} + \omega_3\vec{v}_{31} + \vec{v}_{14} \end{aligned} \tag{6}$$

where we use the shorthand $\vec{v}_{ij}$ to denote the vector $\vec{v}_i - \vec{v}_j$. During the time interval $[0, 1]$, the relative motion $\vec{v} - \vec{v}_4$ along a given direction $\vec{n}$ is

$$(\vec{v} - \vec{v}_4) \cdot \vec{n} = \omega_2\vec{v}_{21} \cdot \vec{n} + \omega_3\vec{v}_{31} \cdot \vec{n} + \vec{v}_{14} \cdot \vec{n} \tag{7}$$

Then, for any point $\vec{x}$ interior to a triangle, we have the following observation if $\vec{v}_4$ lies on the positive side of the triangle.

Observation 4.1 :

1) $(\vec{v} - \vec{v}_4) \cdot \vec{n} > 0$, *if and only if the relative motion makes the triangle move towards the vertex along the direction* $\vec{n}$;

2) $(\vec{v}-\vec{v}_4)\cdot\vec{n}=0$, *if and only if the triangle and the vertex have zero relative motion along the direction* $\vec{n}$;
3) $(\vec{v}-\vec{v}_4)\cdot\vec{n}<0$, *if and only if the relative motion makes the triangle move away from the vertex along the direction* $\vec{n}$.

For any $\vec{v}_4$ within the negative half space, we have the opposite results.

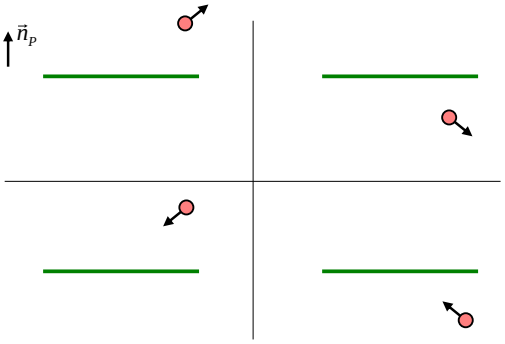


Fig. 2. $\vec{x}_4$'s (red solid circle) position and motion along $\vec{n}_P$ relative to the triangle $\vec{x}_1\vec{x}_2\vec{x}_3$ (green line).

If we refer to Fig. 2-(top row), it is clear from the first statement of Observation 4.1 that the triangle $\vec{x}_1\vec{x}_2\vec{x}_3$ has no chance to collide with the vertex $\vec{x}_4$ if $\vec{x}_4$ lies on one side of the plane and $\vec{x}_1\vec{x}_2\vec{x}_3$ moves to the other side of the plane. Thus, we obtain the second observation:

Observation 4.2 :
A deforming triangle $\vec{x}_1\vec{x}_2\vec{x}_3$ *does not move close to a moving vertex* $\vec{x}_4$ *during the time interval* $[0,1]$ *if*

1) $d_P>0$ *and* $(\vec{v}-\vec{v}_4)\cdot\vec{n}_P\le 0$ *or*
2) $d_P<0$ *and* $(\vec{v}-\vec{v}_4)\cdot\vec{n}_P\ge 0$

where $\vec{n}_P$ *is the normal of the plane containing the triangle* $\vec{x}_1\vec{x}_2\vec{x}_3$.

This observation implicitly gives us a sufficient condition in which a deforming triangle does not collide with a moving vertex.

In addition, as shown in Fig. 2-(bottom row), if $d_P>0$ and $(\vec{v}-\vec{v}_4)\cdot\vec{n}_P>0$, or $d_P<0$ and $(\vec{v}-\vec{v}_4)\cdot\vec{n}_P<0$, the deforming triangle $\vec{x}_1\vec{x}_2\vec{x}_3$ moves close to the vertex $\vec{x}_4$. However, conservative advancement states that the relative motion in the given direction $\vec{n}_P$ must cover the distance d_P in order to bring the two features into contact. Otherwise, the features remain collision-free. Thus, we obtain the third observation based on conservative advancement:

Observation 4.3 :
A deforming triangle $\vec{x}_1\vec{x}_2\vec{x}_3$ *moves close to a moving vertex* $\vec{x}_4$, *but does not come into contact if*

1) $d_P>0$ *and* $0<(\vec{v}-\vec{v}_4)\cdot\vec{n}_P<d_P$ *or*
2) $d_P<0$ *and* $d_P<(\vec{v}-\vec{v}_4)\cdot\vec{n}_P<0$

4.3 Bounds of Relative Motion

We now describe how we compute the bounds of relative motion (Eq. 7) in order to derive our culling conditions. As the relative motion $(\vec{v}-\vec{v}_4)\cdot\vec{n}$ varies with barycentric coordinates ω_2 and ω_3, we introduce a function $f_{\vec{n}}(\omega_2,\omega_3)$ that determines the motion bound as follows

$$\begin{aligned} f_{\vec{n}}(\omega_2,\omega_3) &= (\vec{v}-\vec{v}_4)\cdot\vec{n} \\ &= \omega_2\vec{v}_{21}\cdot\vec{n}+\omega_3\vec{v}_{31}\cdot\vec{n}+\vec{v}_{14}\cdot\vec{n} \end{aligned} \quad (8)$$

Recall that each vertex undergoes a linear motion, i.e. $\vec{x}_i$ has constant velocity $\vec{v}_i$ [22], [23], [24]. We intend to maximize and minimize $f_{\vec{n}}(\omega_2,\omega_3)$ subject to the constraints shown in Fig. 3(left)

$$\begin{aligned} 0\le\omega_2\le 1 \\ 0\le\omega_3\le 1 \\ 0\le 1-\omega_2-\omega_3\le 1 \end{aligned}$$

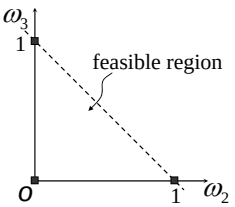


Fig. 3. Bounded Feasible Region of $f_{\vec{n}}(\omega_2,\omega_3)$

TABLE 1
$f_{\vec{n}}(\omega_2,\omega_3)$ Extreme Values

Extreme Points (ω_2,ω_3)	$f_{\vec{n}}(\omega_2,\omega_3)$
(0,0)	$(\vec{v}_1-\vec{v}_4)\cdot\vec{n}$
(1,0)	$(\vec{v}_2-\vec{v}_4)\cdot\vec{n}$
(0,1)	$(\vec{v}_3-\vec{v}_4)\cdot\vec{n}$

This problem can be solved by linear programming, which states that an objective function attains a maximum and a minimum value at extreme points of the feasible region [34]. In our case, the feasible region and extreme points are shown in Fig. 3(right). We calculate the objective function $f_{\vec{n}}(\omega_2,\omega_3)$ at each of these extreme points and tabulate the results as Table 1.

We see that the objective function $f_{\vec{n}}(\omega_2,\omega_3)$ attains a maximum and a minimum at

$$f_{\vec{n}}^{max}=\max_{i=1,2,3}(\vec{v}_i-\vec{v}_4)\cdot\vec{n} \quad (9)$$

and

$$f_{\vec{n}}^{min}=\min_{i=1,2,3}(\vec{v}_i-\vec{v}_4)\cdot\vec{n} \quad (10)$$

Thus we have the bounds of relative motion

$$f_{\vec{n}}^{min}\le(\vec{v}-\vec{v}_4)\cdot\vec{n}\le f_{\vec{n}}^{max} \quad (11)$$

Intuitively speaking, Eq. 11 implies that no other points on the given triangle $\vec{x}_1\vec{x}_2\vec{x}_3$ move faster (or slower) towards the vertex $\vec{v}_4$ than the vertex that realizes the upper bound (or the lower bound) of relative motion. Moreover, we note that Eq. 9 is equivalent to the one in [6].

4.4 Non-Colliding Conditions

We now derive our culling conditions for the face-vertex CA test. By considering Observation 4.2 and the relative motion bounds as obtained above, we have:

Lemma 4.1 *A deforming triangle $\vec{x}_1\vec{x}_2\vec{x}_3$ will not collide with a moving vertex during the time interval [0,1] if*

1) $d_P > 0$ *and* $f^{max}_{\vec{n}_P} \leq 0$ *or*
2) $d_P < 0$ *and* $f^{min}_{\vec{n}_P} \geq 0$

where $\vec{n}_P$ is the normal of the plane containing the triangle $\vec{x}_1\vec{x}_2\vec{x}_3$.

Proof: Since $(\vec{v} - \vec{v}_4) \cdot \vec{n}_P \leq f^{max}_{\vec{n}_P} \leq 0$ and $d_P > 0$, based on the first statement of Observation 4.2, it immediately yields to the first condition. Similarly, since $(\vec{v} - \vec{v}_4) \cdot \vec{n}_P \geq f^{min}_{\vec{n}_P} \geq 0$ and $d_P < 0$, it yields to the second condition based on the second statement of Observation 4.2. □

Failure of the conditions in Lemma 4.1 indicates that the triangle and the vertex move close to each other in the given direction (see Fig. 2-(bottom row)). Using the claim of conservative advancement (Observation 4.3), if the motion along the given direction does not exceed the distance d_P, they have no chance to collide. Thus, we have the following lemma:

Lemma 4.2 *A deforming triangle $\vec{x}_1\vec{x}_2\vec{x}_3$ will not collide with a moving vertex during the time interval [0,1] if*

1) $d_P > 0$, $f^{min}_{\vec{n}_P} > 0$ *and* $f^{max}_{\vec{n}_P} < d_P$ *or*
2) $d_P < 0$, $f^{max}_{\vec{n}_P} < 0$ *and* $f^{min}_{\vec{n}_P} > d_P$

Proof: This directly follows from the definition of conservative advancement (see Observation 4.3). □

From Lemmas 4.1 and 4.2, we have the following corollary:

Corollary 4.3 *A deforming triangle $\vec{x}_1\vec{x}_2\vec{x}_3$ will not collide with a moving vertex during the time interval [0,1] if*

1) $d_P > 0$ *and* $f^{max}_{\vec{n}_P} < d_P$ *or*
2) $d_P < 0$ *and* $f^{min}_{\vec{n}_P} > d_P$

This is our first culling condition for the face-vertex CA test, which if satisfied, proves that there exists no collision between the face and vertex during the motion. Otherwise, it is inconclusive and we then check if the projection $\vec{x}^p_4$ collides with a bounding circle of the triangle $\vec{x}_1\vec{x}_2\vec{x}_3$ (see Fig. 1). Here, we have our second culling condition for face-vertex CA tests.

Lemma 4.4 *A deforming triangle $\vec{x}_1\vec{x}_2\vec{x}_3$ will not collide with a moving vertex $\vec{x}_4$ during the time interval [0,1], if $f^{max}_{\vec{n}_C} < (d_C - r)$, where $\vec{n}_C$ denotes the vector from $\vec{c}$ to $\vec{x}^p_4$ and d_C is the closest distance between $\vec{c}$ and $\vec{x}^p_4$.*

Proof: It directly follows from Proposition 3.2. Please refer to Fig. 1 and Eq. (2) for notation. Here, $f^{max}_{\vec{n}_C}$ is an upper bound of relative motion along the horizontal direction ($\vec{n}_C$). □

If none of the conditions are satisfied, we perform the face-vertex CA to find the time of contact as suggested in [6].

5 Edge-Edge Culling

For the edge-edge case, we first check if the lines containing the two edges collide with each other. Based on Proposition 3.3, we present a culling condition which, if satisfied, proves that the two deforming edges does not collide during the motion. Otherwise, we check if the bounding circles of the two edges' projection overlap based on Proposition 3.4. Similar to the case of face-vertex culling, we present some preliminaries first.

5.1 Closest Distance

As depicted in Fig. 4-(left), let $\vec{x}_1\vec{x}_2$ and $\vec{x}_3\vec{x}_4$ be two deforming edges. $\vec{n}_P$ denotes the vector perpendicular to both $\vec{x}_1\vec{x}_2$ and $\vec{x}_3\vec{x}_4$, and $\vec{n}_{EE}$ denotes the vector that realizes the minimum distance d_{EE} from the edge $\vec{x}_1\vec{x}_2$ to the edge $\vec{x}_3\vec{x}_4$. Let d_P be the distance between the two lines containing $\vec{x}_1\vec{x}_2$ and $\vec{x}_3\vec{x}_4$; then d_P and $\vec{n}_P$ between the two lines can be calculated as

$$d_P = \vec{x}_{41} \cdot \vec{n}_P, \quad \text{where} \quad \vec{n}_P = \frac{\vec{x}_{21} \times \vec{x}_{43}}{|\vec{x}_{21} \times \vec{x}_{43}|} \tag{12}$$

Similar to the Face-Vertex case, d_P is a signed distance and its sign indicates where $\vec{x}_3\vec{x}_4$ lies relative to $\vec{x}_1\vec{x}_2$:

1) if $d_P > 0$, $\vec{x}_3\vec{x}_4$ lies within the half space to which $\vec{n}_P$ points;
2) if $d_P = 0$, $\vec{x}_1\vec{x}_2$ and $\vec{x}_3\vec{x}_4$ lies on the same plane;
3) if $d_P < 0$, $\vec{x}_3\vec{x}_4$ lies within the other half space.

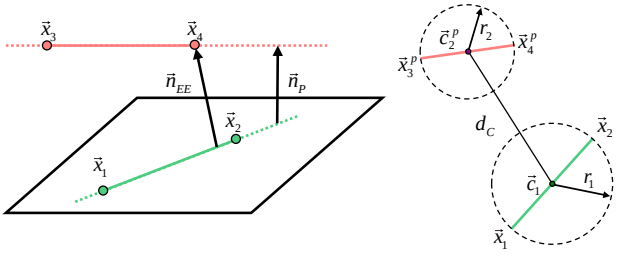


Fig. 4. Edge-Edge Case. First check if the lines containing two edges collide and then check if the two bounding circles collide on the same plane.

As depicted in Fig. 4-(right), when we project the edge $\vec{x}_3\vec{x}_4$ onto the plane containing the edge $\vec{x}_1\vec{x}_2$, we obtain a projected edge $\vec{x}^p_3\vec{x}^p_4$. Let $\vec{c}_1$ be the middle point of $\vec{x}_1\vec{x}_2$ and let $\vec{c}^p_2$ be the middle point of $\vec{x}^p_3\vec{x}^p_4$. Then the distance between the two middle points can be calculated as

$$d_C = |\vec{c}^p_2 - \vec{c}_1| \quad \text{where} \quad \vec{c}^p_2 = \vec{c}_2 - d_P\vec{n}_P \tag{13}$$

where $\vec{c}_2$ is the middle point of $\vec{x}_3\vec{x}_4$.

5.2 Relative Edge-Edge Motion and Bounds

Similarly to the face-vertex case, each edge is assumed to undergo a linear motion. The relative velocity between $\vec{x}$ on the edge $\vec{x}_1\vec{x}_2$ and $\vec{x}'$ on the edge $\vec{x}_3\vec{x}_4$ is

$$\begin{aligned}\vec{v} - \vec{v}' &= \omega_1\vec{v}_1 + (1-\omega_1)\vec{v}_2 - \omega_2\vec{v}_3 - (1-\omega_2)\vec{v}_4 \\ &= \omega_1\vec{v}_{12} + \omega_2\vec{v}_{43} + \vec{v}_{24}\end{aligned}$$

Like before, we introduce a function $g_{\vec{n}}(\omega_1, \omega_2)$ to be the relative motion $\vec{v} - \vec{v}'$ along a given direction $\vec{n}$

$$g_{\vec{n}}(\omega_1, \omega_2) = \omega_1\vec{v}_{12}\cdot\vec{n} + \omega_2\vec{v}_{43}\cdot\vec{n} + \vec{v}_{24}\cdot\vec{n} \qquad (14)$$

Similarly to the face-vertex case (see Section 4.3), a maximum and a minimum can be attained as:

$$g_{\vec{n}}^{max} = \max_{i,j=1,2}(\vec{v}_i - \vec{v}_j)\cdot\vec{n} \qquad (15)$$

and

$$g_{\vec{n}}^{min} = \min_{i,j=1,2}(\vec{v}_i - \vec{v}_j)\cdot\vec{n} \qquad (16)$$

Intuitively speaking, Eqs. 15 and 16 imply that no other pairs of points on the two edges $\vec{x}_1\vec{x}_2$ and $\vec{x}_3\vec{x}_4$ move towards each other faster (or slower) than the vertex pair that realizes the upper bound (or the lower bound) of relative motion. We note that Eq. 15 yields a result equivalent to [6].

5.3 Non-Colliding Conditions

We now give our culling conditions for the edge-edge CA tests analogous to the ones in Section 4.4.

Lemma 5.1 *A deforming edge $\vec{x}_1\vec{x}_2$ will not collide with another deforming edge $\vec{x}_3\vec{x}_4$ during the time interval [0,1] if*

1) $d_P > 0$ *and* $g_{\vec{n}_P}^{max} < d_P$ *or*
2) $d_P < 0$ *and* $g_{\vec{n}_P}^{min} > d_P$

If any condition in Lemma 5.1 is satisfied, it proves that there exists no collision between the feature pair (i.e. edge-edge) during the motion. Otherwise, we may check the next condition below:

Lemma 5.2 *A deforming edge $\vec{x}_1\vec{x}_2$ will not collide with another deforming edge $\vec{x}_3\vec{x}_4$ during the time interval [0,1], if $g_{\vec{n}_C}^{max} < (d_C - r_1 - r_2)$, where $\vec{n}_C$ denotes the vector from $\vec{c}_1$ to $\vec{c}_2^{\,p}$, d_C is the closest distance between $\vec{c}_1$ and $\vec{c}_2^{\,p}$, and r_i is the radius of a bounding circle of the edge at $t = 0$.*

If none of the conditions are satisfied, we perform the edge-edge CA to find the time of contact as suggested in [6]. We can prove Lemmas 5.1 and 5.2 similarly to Lemmas and Corollaries 4.1 $\sim$ 4.4, and here we omit their proofs.

6 Triangle-Level Culling

To perform CCD between two deforming triangles, in principle, we need 6 face-vertex and 9 edge-edge elementary CCD tests. However, the following corollaries will state that it is not even necessary to consider all these 15 elementary tests. By reusing the culling results from the face-vertex and edge-edge tests explain in the earlier sections, our algorithm avoids unnecessary elementary tests and even can terminate as soon as some conditions are satisfied.

6.1 Cullings based on Face-Vertex Tests

The following corollary states if the two vertices of an edge satisfy the same condition (either statement 1 or 2) of Corollary 4.3, the edge proves to be collision-free from a triangle.

Corollary 6.1 *An edge $\vec{x}_1\vec{x}_2$ out of a deforming triangle will not collide with another deforming triangle during the time interval [0,1] if*

$$\begin{aligned}1)\quad & d_{P,\vec{x}_1} > 0, \quad f_{\vec{n}_P,\vec{x}_1}^{max} \le d_{P,\vec{x}_1} \\ & d_{P,\vec{x}_2} > 0, \quad f_{\vec{n}_P,\vec{x}_2}^{max} \le d_{P,\vec{x}_2}\end{aligned}$$

or

$$\begin{aligned}2)\quad & d_{P,\vec{x}_1} < 0, \quad f_{\vec{n}_P,\vec{x}_1}^{min} \ge d_{P,\vec{x}_1} \\ & d_{P,\vec{x}_2} < 0, \quad f_{\vec{n}_P,\vec{x}_2}^{min} \ge d_{P,\vec{x}_2}\end{aligned}$$

where $\vec{n}_P$ is the normal of the second triangle; $d_{P,\vec{x}_1}$ and $d_{P,\vec{x}_2}$ are the distances from $\vec{x}_1$ and $\vec{x}_2$ to the plane, respectively.

Proof: see [35].

This corollary implies that we can cull three edge-edge tests (i.e. $\vec{x}_1\vec{x}_2$ vs. three edges out of the second triangle). Based on Corollary 6.1, we now have Corollary 6.2:

Corollary 6.2 *A deforming triangle $\vec{x}_1\vec{x}_2\vec{x}_3$ will not collide with another deforming triangle during the time interval [0,1], if*

$$\begin{aligned}1)\quad & d_{P,\vec{x}_1} > 0, \quad f_{\vec{n}_P,\vec{x}_1}^{max} \le d_{P,\vec{x}_1} \\ & d_{P,\vec{x}_2} > 0, \quad f_{\vec{n}_P,\vec{x}_2}^{max} \le d_{P,\vec{x}_2} \\ & d_{P,\vec{x}_3} > 0, \quad f_{\vec{n}_P,\vec{x}_3}^{max} \le d_{P,\vec{x}_3}\end{aligned}$$

or

$$\begin{aligned}2)\quad & d_{P,\vec{x}_1} < 0, \quad f_{\vec{n}_P,\vec{x}_1}^{min} \ge d_{P,\vec{x}_1} \\ & d_{P,\vec{x}_2} < 0, \quad f_{\vec{n}_P,\vec{x}_2}^{min} \ge d_{P,\vec{x}_2} \\ & d_{P,\vec{x}_3} < 0, \quad f_{\vec{n}_P,\vec{x}_3}^{min} \ge d_{P,\vec{x}_3}\end{aligned}$$

where $\vec{n}_P$ is the normal of the second triangle; $d_{P,\vec{x}_1}$ $d_{P,\vec{x}_2}$ and $d_{P,\vec{x}_3}$ are the distances from $\vec{x}_1$, $\vec{x}_2$ and $\vec{x}_3$ to the plane containing the second triangle, respectively.

6.2 Cullings based on Edge-Edge Tests

Based on the result of the edge-edge test, we can use Corollary 6.3 to cull two additional edge-edge pairs (i.e. $\vec{x}_1\vec{x}_2$ against the other two edges out of the second triangle $\vec{x}'_2\vec{x}'_3$ and $\vec{x}'_3\vec{x}'_1$) and two face-vertex pairs (i.e. $\vec{x}_1$ and $\vec{x}_2$ against the second triangle if they are survived from the previous culling stages).

Under the assumption that the edge $\vec{x}'_1\vec{x}'_2$ and the vertex $\vec{x}'_3$ lie on the same side of a plane containing $\vec{x}_1\vec{x}_2$ at $t = 0$ (i.e. $d_P d_{P,\vec{x}'_3} > 0$), the following corollary states that if $\vec{x}_1\vec{x}_2$ and $\vec{x}'_1\vec{x}'_2$ are collision-free due to Lemma 5.1 , and $\vec{x}'_3$ and the plane containing $\vec{x}_1\vec{x}_2$ are also collision-free due to Corollary 4.3, then $\vec{x}_1\vec{x}_2$ will not collide with a triangle $\vec{x}'_1\vec{x}'_2\vec{x}'_3$.

Corollary 6.3 *Given an edge $\vec{x}_1\vec{x}_2$ of a deforming triangle and an edge $\vec{x}'_1\vec{x}'_2$ of another deforming triangle, $\vec{x}_1\vec{x}_2$ will not collide with the second triangle if*

$$1) \quad \begin{array}{rcl} d_P > 0 & and & g^{max}_{\vec{n}_P} \le d_P \\ d_{P,\vec{x}'_3} > 0 & and & f^{max}_{\vec{n}_P,\vec{x}'_3} \le d_{P,\vec{x}'_3} \end{array}$$

or

$$2) \quad \begin{array}{rcl} d_P < 0 & and & g^{min}_{\vec{n}_P} \ge d_P \\ d_{P,\vec{x}'_3} < 0 & and & f^{min}_{\vec{n}_P,\vec{x}'_3} \ge d_{P,\vec{x}'_3} \end{array}$$

where $\vec{n}_P$ is the normal of the plane spanned by the two edges, d_P is the distances between two lines containing the two edges, $g^{max}_{\vec{n}_P}$ and $g^{min}_{\vec{n}_P}$ are the motion bounds of $\vec{x}_1\vec{x}_2$ against $\vec{x}'_1\vec{x}'_2$ along the direction $\vec{n}_P$, $d_{P,\vec{x}'_3}$ is the distance from $\vec{x}'_3$ to the plane, and $f^{max}_{\vec{n}_P,\vec{x}'_3}$ and $f^{min}_{\vec{n}_P,\vec{x}'_3}$ are the motion bounds of $\vec{x}'_3$ against the plane along the direction $\vec{n}_P$.

Proof: see [35].

7 PUTTING ALL TOGETHER

We have described different culling tests, and now discuss the sequence in which these culling tests should be executed.

Given two deforming triangles, we first perform face-vertex culling tests for the vertex out of the first triangle against the second triangle using Corollary 4.3. If satisfied, it proves that there does not exist any collision between the vertex and the triangle, and at the same time we report the sign of d_P of the vertex with respect to the triangle.

$$\text{sign}(d_P) = \begin{cases} 1 & d_P > 0 \text{ and } f^{max}_{\vec{n}_P} < d_P \\ -1 & d_P < 0 \text{ and } f^{min}_{\vec{n}_P} > d_P \\ 0 & \text{otherwise} \end{cases}$$

With these signs, we can perform further culling tests based on Corollary 6.2. If three vertices have the same sign, we terminate the whole algorithm and report the absence of collision between the two deforming triangles. Otherwise, we perform the same culling test for each vertex out of the second triangle against the first triangle using Corollary 4.3 and do the further culling with Corollary 6.2.

Second, using Corollary 6.1, we check if both the end vertices of an edge out of the first triangle have the same sign, which, if satisfied, prove there does not exist any collision between the edge and the second triangle. Thus, we can cull three more edge-edge pairs. We then perform the same sign checking for an edge out of the second triangle against the first triangle.

Third, for the edge-edge pairs survived from the previous culling stages, we perform the edge-edge culling test using Lemma 5.1. If the condition is satisfied, there is no collision between the two edges and the sign of d_P is reported. For a parallel case, we simply let $d_P = 0$ and leave it to the next test. Followed by every edge-edge culling test, we perform the face-vertex culling test for the vertex opposite to the second edge on the same triangle against the face containing the first edge. If the sign reported by the edge-edge culling test is the same as the one reported by the face-vertex culling test, we can additionally cull two more edge-edge pairs and two face-vertex pairs based on Corollary 6.3.

Fourth, we may optionally perform face-vertex culling tests using Lemma 4.4 and edge-edge culling tests using Lemma 5.2.

Finally, we perform the conservative advancement for those face-vertex and edge-edge pairs that survived all the culling stages explained so far.

We summarize the sequence of all the culling tests and their priorities used in our culling pipeline (see Table 2). In [35], we provide the pseudocodes of some important culling steps such as sequences 1-4.

TABLE 2
Culling Sequence and Priority

Sequence	Culling steps	Priority
1	Corollary 4.3	necessary
2	Corollary 6.2	necessary
3	Corollary 6.1	necessary
4	Lemma 5.1	necessary
5	Corollary 6.3	optional
6	Lemma 4.4	optional
7	Lemma 5.2	optional

8 IMPLEMENTATION RESULTS

We have implemented our algorithm using C++ language under Windows XP. The pseudocodes are also given in Appendix. Though efficient, our implementation is very simple.

We now present some comparative statistics and experimental results. We first provide the performance comparison of distance calculation for different cases. Then we show the performance improvement of our approach against existing algorithms.

8.1 Comparisons on Distance Computation

We randomly generate a triangle and a point, and measure the timings when computing distance be-

tween 1) the plane containing the triangle and the point (plane-vertex); 2) the triangle and the point (face-vertex). We use the code provided in [36] for computing the distance between a triangle and a point.

Similarly, we randomly generate two line segments, and measure the timings when computing distance between 1) the two lines containing the line segments (line-line); 2) the two line segments (edge-edge). We use the code provided in SWIFT++ [37] for computing the distance between two line segments.

We repeat the random procedure for 1000 times and show the costs of distance computation in Fig. 5 and Fig. 6. The timings were measured on a desktop PC with 2.67Hz Intel Core i7 CPU and 3.0G memory. As shown in Fig. 5, the cost of computing the distance between a face and a vertex is about 9.6 times higher than that between a plane and a vertex. Meanwhile, as shown in Fig. 6, the cost of computing the distance between two line segments about $1.7 \sim 3.0$ times higher than that between two lines. Naturally, the cost of computing the distance between two line segments varies depending on the configuration of two lines.

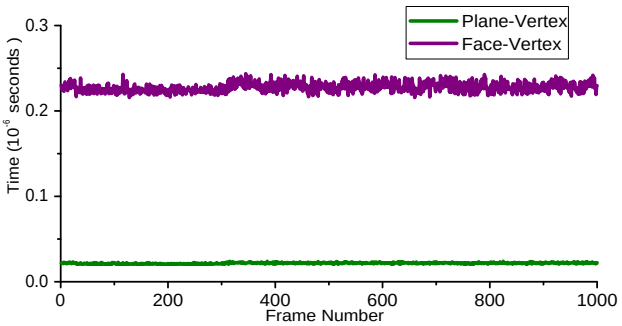


Fig. 5. Distance Computation Costs: Plane-Vertex vs Face-Vertex.

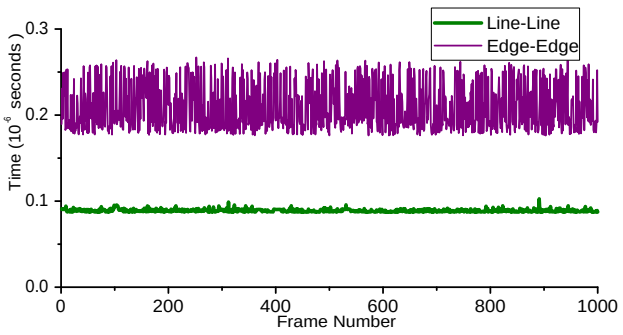


Fig. 6. Distance Computation Costs: Line-Line vs Edge-Edge.

Such experimental statistics and comparison provide further insight to the effectiveness of our culling approach - utilizing computationally cheaper operations as much as possible prior to performing relatively expensive full conservative advancement. These results also indicate the sequence in which the culling procedures should be executed. Since distance computation in the case of plane-vertex is cheaper than the case of line-line, we first perform the culling procedures associated with the former (see Section 7).

8.2 Comparisons on CCD Performance

We now show the performance improvement of our culling approach over existing algorithms. In our first experiment, we randomly generate a thousand pairs of non-colliding deforming triangles and measure the performance of our CA algorithm. We also compare it with other methods including a straighforward feature-level CA algorithm, a triangle-level CA algorithm [6], and an algebraic approach using cubics [20], [27] (see Fig. 7). Here, the straighforward feature-level CA test includes 15 elementary CA tests: six face-vertex and nine edge-edge CA tests. Moreover, to reduce the number of CA iterations for the straighforward algorithm, temporal culling [4] has been integrated. In this experiment, our approach can provide up to 30 times speedup compared to the straighforward algorithm, since most face-vertex and edge-edge pairs can be culled in the early stage (i.e. face-vertex culling test). In addition, our approach can provide up to $5 \sim 6$ times speedup in comparison with the triangle-level CA. We also show the performance of a cubic solver implemented in OpenCCD [20] as shown in pink in Fig. 7, and our approach outperforms it by a factor of 20.

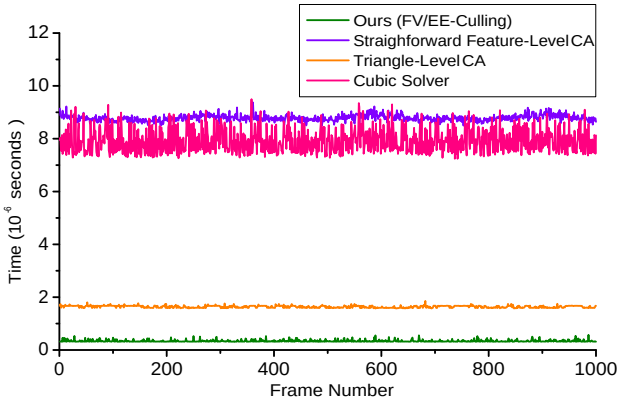


Fig. 7. Performance comparisons using different primitive-level CCD methods for non-colliding deforming triangles. We observe 30 times speedup over a straighforward feature-level CA, $5 \sim 6$ times speedup over a triangle-level CA and 20 times speedup over a cubic solver.

In our second experiment, we randomly generate a pair of deforming triangles which can be, however,

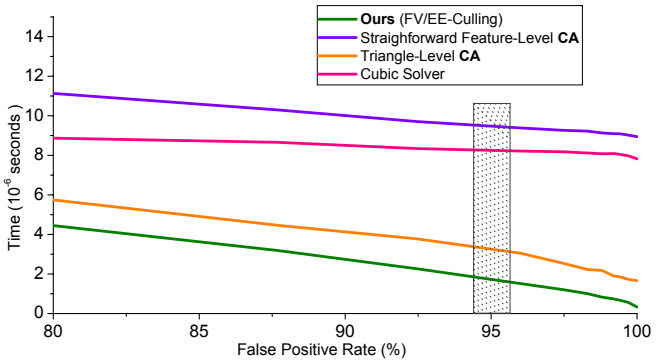


Fig. 8. Computational costs with respect to false positive rates.

either colliding or non-colliding. We repeat this for thousands times and measure the rates of false positive collision results. Then, we measure the average timings of the different primitive-level CCD methods in terms of these false positive rates, as shown in Fig. 8. Existing BVH-based CCD algorithms typically result in a high number of false positives, and as reported in [7], the false positive rates can be as high as, or even higher than 95%. In this case, as highlighted with a box in Fig. 8, our approach outperforms the straightforward feature-level CA by 7 times, the triangle-level CA by 2 times and the OpenCCD cubic solver by 6 times in the range of 95% false positive rates.

8.3 Comparisons using Complicated Models

We have tested our algorithm using the well-known UNC dynamics benchmark[2] consisting of cloth simulation, exploding dragon and N-body deformable simulation. We have integrated our algorithm into OpenCCD[3] to test its performance on these complicated simulation scenes. OpenCCD performs BVH-based collision culling as well as selective BVH restructuring [20] to cope with model deformation, and our algorithm replaces the elementary CCD test in OpenCCD, the cubic solver. Fig. 9 shows snapshots during the cloth simulation. Fig. 10 shows the exploding dragon benchmark, in which a bunny model is dropped onto a dragon model and the dragon model breaks into a large number of smaller pieces. Fig. 11 shows an N-body deformable simulation, in which each ball undergoes a rigid or deformable motion and the balls collide with each other and the obstacles.

In addition, in Figs. 9 $\sim$ 11, we compare the performance of our algorithm against that of other existing algorithms, including the straightforward feature-level CA, triangle-level CA and cubic solver. We show only the timings of elementary CCD tests to highlight them. For the cloth simulation, on average, we observe 5.4, 3.2 and 3.8 times speedups over the straightforward feature-level CA, the triangle-level CA and the cubic solver, respectively. We also break down the whole simulation into a few stages that show different performance patterns of decreasing false positive rates. For the exploding dragon, on average, we observe 1.8, 1.3 and 1.5 times speedups over the straightforward feature-level CA, the triangle-level CA and the cubic solver, respectively. In such an exploding scenario, known as a very challenging case for collision detection, the performance improvement is less distinct because of the lack of motion coherence and because of a large number of deep penetration cases. For the N-body deformable simulation, on average, we observe 8.8, 4.3 and 6.3 times speedups over the straightforward feature-level CA, the triangle-level CA and the cubic solver, respectively. We observe that our algorithm is particularly suitable for the scenarios that have high motion coherence and few deep penetration cases. This is because such scenarios are more like to satisfy Propositions 3.1$\sim$3.4.

2. http://gamma.cs.unc.edu/DYNAMICB
3. http://sglab.kaist.ac.kr/OpenCCD

9 Conclusion and Discussion

We have presented a simple, yet efficient culling approach to CA-based continuous collision detection for deforming triangles. Our approach is based on simple culling conditions that prove that there does not exist any collision between a pair of features or triangles. Our culling approach can be incorporated into any CCD algorithms for deformable models that eventually needs elementary CCD tests, including the one based on algebraic equation solvers. For example, before solving 15 cubic equations, one may use our algorithm to cull non-colliding features so as to reduce the number of cubic equations.

In our work, we have assumed that a deformable model has a constant velocity for each vertex, since an interpolating motion with a constant velocity is most widely used in the literature. Nevertheless, the basic culling ideas (i.e. Propositions 3.1$\sim$3.4) can be extended to any non-constant motion, as long as the upper bound of relative motion can be calculated. For example, upper bounds of vertex velocities can be determined using functional analysis or numerical methods, and then an upper bound of relative motion for the features can be computed with these bounds of vertex velocities. We leave this problem as future work.

Acknowledgments

This research work was supported in part by the NRF grant funded by the Korea government (MEST) (No. 2009-0086684) and IT R&D program of MKE/MCST/KOCCA (2008-F-033-02).

REFERENCES

[1] M. Teschner et al., "Collision detection for deformable objects," in *Eurographics*, 2004, pp. 119–139.

[2] B. V. Mirtich, *Impulse-based Dynamic Simulation of Rigid Body Systems*, Ph.D. thesis, University of California, Berkeley, 1996.

[3] X. Y. Zhang, M. Lee, and Y. J. Kim, "Interactive continuous collision detection for non-convex polyhedra," *Pacific Graphics*, pp. 749–760, 2006.

[4] X. Y. Zhang, S. Redon, M. Lee, and Y. J. Kim, "Continuous collision detection for articulated models using Taylor models and temporal culling," *SIGGRAPH*, vol. 26, no. 3, pp. 15, 2007.

[5] M. Tang, Y. J. Kim, and D. Manocha, "C^2A: Controlled conservative advancement for continuous collision detection of polygonal models," *ICRA*, 2009.

[6] M. Tang, Y. J. Kim, and D. Manocha, "Continuous collision detection for non-rigid contact computations using local advancement," in *ICRA*, 2010.

[7] M. Tang, D. Manocha, and R. Tong, "Fast continuous collision detection using deforming non-penetration filters," in *Interactive 3D Graphics and Games*, 2010, pp. 7–13.

[8] J. F. Canny, "Collision detection for moving polyhedra," *IEEE Trans. PAMI*, vol. 8, pp. 200–209, 1986.

[9] Y.-K. Choi, W. Wang, Y. Liu, and M.-S. Kim, "Continuous collision detection for elliptic disks," *IEEE Trans. Robotics*, 2006.

[10] B. Kim and J. Rossignac, "Collision prediction for polyhedra under screw motions," in *ACM SMA*, 2003.

[11] S. Redon, A. Kheddar, and S. Coquillart, "An algebraic solution to the problem of collision detection for rigid polyhedral objects," *ICRA*, 2000.

[12] K. Abdel-Malek, D. Blackmore, and K. Joy, "Swept volumes: Foundations, perspectives, and applications," *Journal of Shape Modeling*, 2002.

[13] S. Cameron, "Collision detection by four-dimensional intersection testing," *IEEE Trans. Robotics and Automation*, vol. 6, pp. 291–302, 1990.

[14] S. Redon, A. Kheddar, and S. Coquillart, "Fast continuous collision detection between rigid bodies," *Eurographics*, 2002.

[15] F. Schwarzer, M. Saha, and J.-C. Latombe, "Exact collision checking of robot paths," in *WAFR*, 2002.

[16] P. K. Agarwal, J. Basch, L. J. Guibas, J. Hershberger, and L. Zhang, "Deformable free space tiling for kinetic collision detection," in *WAFR*, 2001, pp. 83–96.

[17] D. Kim, L. Guibas, and S. Shin, "Fast collision detection among multiple moving spheres.," *IEEE Trans. Vis. Comput. Graph.*, vol. 4, no. 3, pp. 230–242, 1998.

[18] D. Kirkpatrick, J. Snoeyink, and Bettina Speckmann, "Kinetic collision detection for simple polygons," in *ACM Computational Geometry*, 2000, pp. 322–330.

[19] G. van den Bergen, "Ray casting against general convex objects with application to continuous collision detection," *Journal of Graphics Tools*, 2004.

[20] D. Kim, J.-P. Heo, J. Huh, J. Kim, and S.-E. Yoon, "HPCCD: Hybrid parallel continuous collision detection," *Pacific Graphics*, 2009.

[21] S. Redon, "Continuous collision detection," in *Haptic Rendering: Foundations, Algorithms and Applications*, 2009.

[22] M. Moore and J. Wilhelms, "Collision detection and response for computer animation," in *SIGGRAPH*, 1988, pp. 289–298.

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

[24] R. Bridson, R. Fedkiw, and J. Anderson, "Robust treatment of collisions, contact and friction for cloth animation," *SIGGRAPH*, 2002.

[25] S. Curtis, R. Tamstorf, and D. Manocha, "Fast collision detection for deformable models using representative-triangles," *Interactive 3D Graphics and Games*, pp. 61–79, 2008.

[26] M. Tang, S. E. Yoon, and D. Manocha, "Adjacency-based culling for continuous collision detection," *Visual Computers*, vol. 24, pp. 545–553, 2008.

[27] M. Tang, S. Curtis, S. E. Yoon, and D. Manocha, "Interactive continuous collision detection between deformable models using connectivity-based culling," *ACM SPM*, pp. 25–36, 2008.

[28] N.K. Govindaraju, D. Knott, N. Jain, I. Kabul, R. Tamstorf, R. Gayle, M.C. Lin, and D. Manocha, "Interactive collision detection between deformable models using chromatic decomposition," *SIGGRAPH*, vol. 24, no. 3, pp. 991–999, 2005.

[29] W. Wong and G. Baciu, "Robust continuous collision detection for interactive deformable surfaces," *Journal of Computer Animation and Virtual Worlds*, 2007.

[30] J. Barbic and D. L. James, "Subspace self-collision culling," *SIGGRAPH*, vol. 29, no. 3, 2010.

[31] S. C. Schvartzman, . G. Pérez, and M. A. Otaduy, "Star-contours for efficient hierarchical self-collision detection," *SIGGRAPH*, vol. 29, no. 3, 2010.

[32] N.K. Govindaraju, S. Redonn, M.C. Lin, and D. Manocha, "Quick-CULLIDE: Efficient inter- and intra-object collision culling using graphics hardware," in *VR*, 2005, pp. 59–66,319.

[33] N. K. Govindaraju, I. Kabul, M. Lin, and D. Manocha, "Fast continuous collision detection among deformable models using graphics processors," *Journal of Computers and Graphics*, vol. 31, pp. 5–14, 2007.

[34] H. Anton and C. Rorres, *Elementary Linear Algebra: Applications Version (10th Ed.)*, Wiley, 2010.

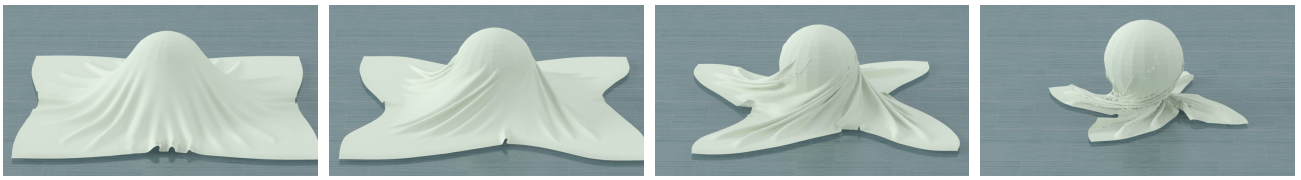
[35] X. Y. Zhang and Y. J. Kim, "Proofs and pseudocodes for simple culling methods for continuous collision detection of deforming triangles," Tech. Rep. CSE–TR-2011-01, Ewha Womans University, Korea, 2011.

[36] Philip J. S. and David H. E., *Geometric Tools for Computer Graphics*, 2002.

[37] S. Ehmann and M. C. Lin, "Accurate and fast proximity queries between polyhedra using convex surface decomposition," *Eurographics*, vol. 20, no. 3, pp. 500–510, 2001.

Xinyu Zhang is a research professor of computer science and engineering at Ewha Womans University. He was a full-time lecturer (2007-2008) and a postdoctoral research fellow (2005-2007) in the Department of Computer Science and Engineering at Ewha Womans University. He obtained his PhD degree in Computer Science, MS and BS degrees in Material Science from Zhejiang University in 2004, 2000 and 1997, respectively. His research interests include computer graphics, geometric modeling, and collision detection.

Young J. Kim is an associate professor of computer science and engineering at Ewha Womans University. He received his PhD in computer science in 2000 from Purdue University. Before joining Ewha, he was a postdoctoral research fellow in the Department of Computer Science at the University of North Carolina at Chapel Hill. His research interests include interactive computer graphics, computer games, robotics, haptics, and geometric modeling. He has published more than 50 papers in leading conferences and journals in these fields. He also received the best paper awards at the ACM Solid Modeling Conference in 2003 and the International CAD Conference in 2008, and the best poster award at the Geometric Modeling and Processing conference in 2006. He was selected as best research faculty of Ewha in 2008, and received the outstanding research cases award from Korean research foundation in 2008.



(a) Frame 90 (b) Frame 270 (c) Frame 320 (d) Frame 450

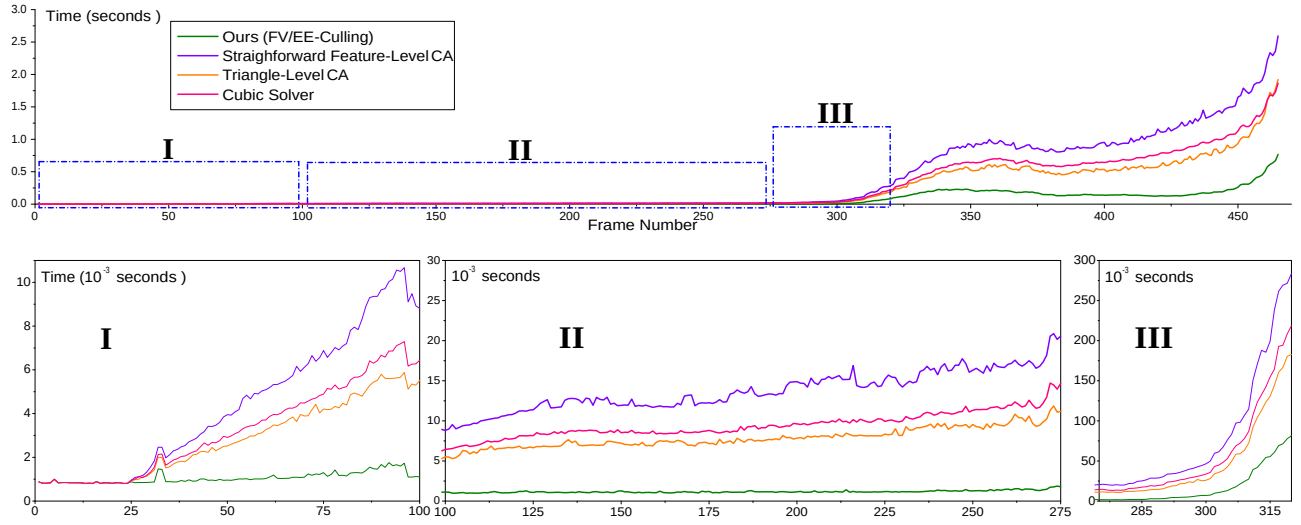


Fig. 9. Cloth Benchmark. **Top**: a piece of cloth is dropped onto the top of a ball and the ball is twisted to generate complex folds and wrinkles on the cloth. This model consists of 46,598 vertices and 92,230 triangles and the simulation results in a high number of self-collisions and close contacts. **Middle**: performance comparisons using different primitive-level CCD methods. **Bottom**: Zoom-in views.

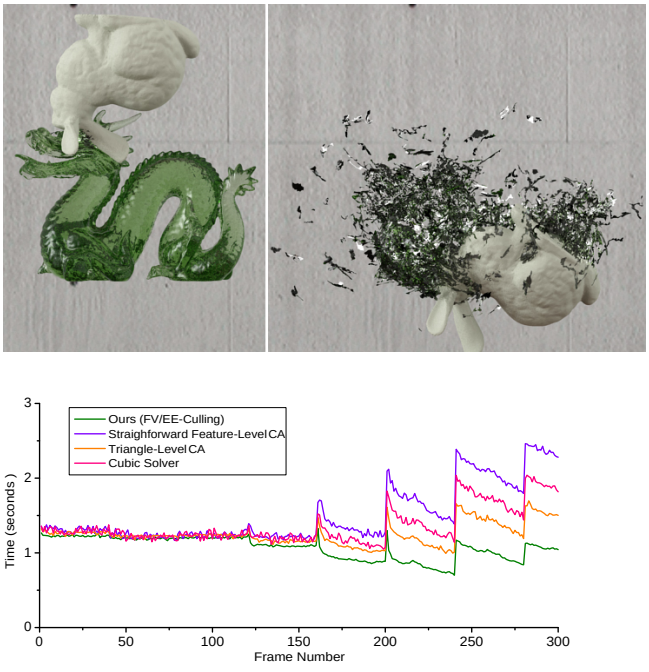


Fig. 10. **Top**: Exploding dragon: a bunny is dropped onto a dragon and the dragon breaks into a large number of smaller pieces. **Bottom**: Performance comparisons using different primitive-level CCD methods.

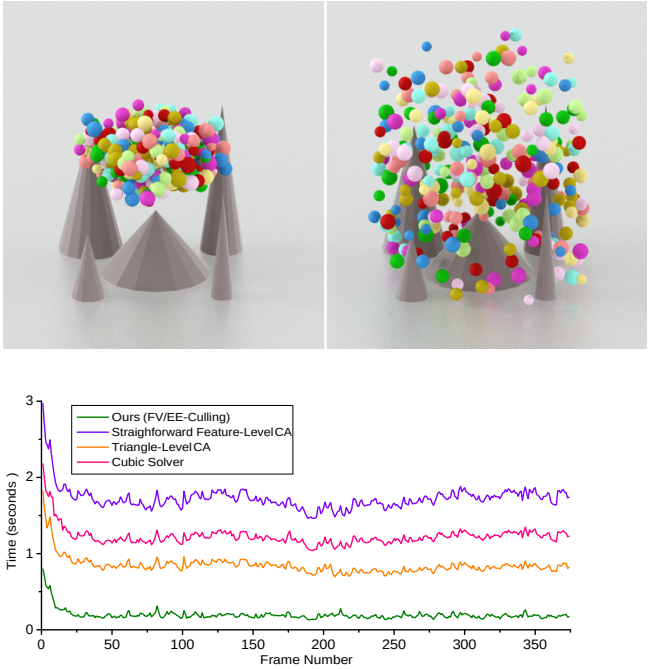


Fig. 11. **Top**: N-Body Simulation. Each ball undergoes a rigid or deformable motion. **Bottom**: Performance comparisons using different primitive-level CCD methods.