Fast and Robust Continuous Collision Detection (fastCCD)

DreamWorks Animation Technical Report 2014-320

Suejung Huh
Ohhh! Inc.

Young Ju Lee
Texas State University

Galen Gornowicz
DreamWorks Animation

Ronald D. Henderson
DreamWorks Animation

Abstract

We present a fast and robust narrow-phase continuous collision detection (CCD) method. Our algorithm is based on the method originally proposed by [Provot 1997]. However, the traditional method has been reformulated analytically so as to obtain more robust collision detection results than other previous CCD methods available, while surpassing the performance of the traditional cubic solver-based methods. The novelties of our method are twofold. First, a new area inclusion test is introduced and analyzed to prove it does not miss a collision, proving our method is robust in terms of *false negatives*. Second, we provide a dimensional reduction technique to handle degenerate cases exhaustively, thereby enabling it to handle such challenging cases without incurring unnecessary computational overhead. Using a number of benchmark tests performed with both randomly generated and publicly available data sets, we demonstrate that our proposed method, *fastCCD*, is both more efficient and more accurate than the existing techniques *exactCCD* [Brochu et al. 2012], *safeCCD* [Wang 2014], and *openCCD* [Kim and Yoon 2009].

CR Categories: I.3.7 [Computer Graphics]: Three-dimensional Graphics and Realism—Animation

Keywords: cloth simulation, rigid body simulation, continuous collision detection

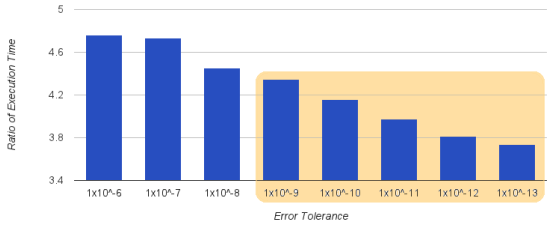

Figure 1: *Comparison of CPU time to execute fastCCD versus exactCCD, where 10 Million random data collision pairs are tested. From an error tolerance of* 10^{-9} *and lower, the collision detection result from fastCCD matches with the one from exactCCD while fastCCD is significantly faster.*

Table 1: *For* 10 *million random data collision pairs, listed are number of detected collisions and computation time of* FastCCD. *Note that* exactCCD *reports* $749,985$ *collisions in* 37.4 *sec and* safeCCD *reports* $918,663$ *collisions in* 13.6 *sec with* 19% *of false positives. Note that* exactCCD *reports only odd root collisions.*

Error Tolerance	*1e-06*	*1e-07*	*1e-08*	*1e-09*	*1e-10*	*1e-11*	*1e-12*	*1e-13*
Time (in sec)	7.85	7.9	8.396	8.6	9	9.4	9.8	10
Odd Collisions	750127	750005	749987	749985	749985	749985	749985	749985
Even Collisions	22790	22779	22779	22779	22779	22779	22779	22779

1 Introduction

Continuous collision detection (CCD) is a fundamental technique used in various applications areas, including dynamic simulations and ray tracing in computer graphics and games, and path-finding in artificial intelligence. In general, CCD is performed in two phases: a broad phase and a narrow phase [Mirtich 1997a]. Given n pairs of collision candidates, the naive CCD approach would induce $O(n^2)$ CCD operations [Hahn 1988; Moore and Wilhems 1988]. In the broad phase, the potential colliding pairs are culled. There are numerous research papers on culling methods such as sweep and prune [Witkin and Baraff 1997], hierarchical spatial subdivision [Mirtich 1997a], non-penetration filters [Tang et al. 2010b] and parallel spatial subdivision [Erleben et al. 2005]. Once irrelevant collision pairs have been culled in the broad phase, the actual collision detection is performed for each remaining pair in the narrow phase. In the narrow phase, approximate or *non-exact CCD* methods are typically used in practice. In this paper we focus only on narrow phase continuous collision detection.

The narrow phase of CCD has been one of the performance bottlenecks in large-scale dynamic simulation. The goal for CCD is to design an algorithm that achieves fail-safe collision detection, namely the correct detection of all collisions with minimal computation. In this paper we propose a method which reports exact collision detection results when using a sufficiently small error tolerance, but using a fraction of the computation time needed by other methods. Figure 1 and Table 1 show the performance of our algorithm relative to *exactCCD* and *safeCCD*.

Inaccurate algorithms can miss collisions (*false negatives*) or flag non-colliding cases (*false positives*). In particular, in dynamic simulations, *false negatives* can result in unacceptable simulation results, in which case a node or edge can penetrate into a face or edge, respectively, and from then on, this node or edge stays on a wrong side. This incidence eventually makes an object appear pinched in a simulation. In ray tracing, a *false negative* can cause light slip-through and leaking [Woop et al. 2013]. *False positives* invoke a collision response when there is no actual collision. While the consequences of *false positives* are less severe, excessive *false positives* can cause an unnecessary slowdown in simulations and make a simulation unsuccessful.

Along this line of research, a major advancement was made by [Brochu et al. 2012] who introduced a geometrically exact continuous collision detection method (*exactCCD*). However, their method is computationally expensive. In order to perform geometrically exact collision detection, the method transforms the parametric domain of the time and barycentric coordinates, and does ray casting to check the parity. The proposed method extrapolates a problem to multiple folds as much as the number of faces in the transformed domain. This consequently makes the method slower by an order of magnitude than the other non-exact collision detection methods (see §5 in [Brochu et al. 2012]). In addition, their method inherently ignores the case of nonzero even root count collisions between a point and a face and between two edges. Furthermore, the method inherently reports only whether there is a collision, without providing the time of contact, which is generally needed for resolving collisions [Harmon et al. 2008].

[Wang 2014] proposed *safeCCD* with a set of modifications to basic CCD methods to eliminate false negatives, while still being faster than *exactCCD*. They derive a set of provably safe tolerance settings from a single upper bound B on the relative positions and velocities encountered during simulation. Their proposed modification is to first apply the regular cubic root solver and then to apply additional routines to handle degenerate or nearly degenerate cases exhaustively. They demonstrate that these changes only cause a modest increase in execution time while still running faster than *exactCCD*, but do increase the number of false positives by as much as 10 percent in their experiments.

In this paper, we identify a number of major issues in the traditional CCD cubic solver approach [Provot 1997] and provide remedies, thereby obtaining a more optimal CCD algorithm. Like [Wang 2014], we also consider the problem of how to obtain a comprehensive error bound in coplanarity root-finding, barycentric coordinate calculations, and distance calculations. In comparison, we have obtained a tighter error bound than *safeCCD*, which limits the introduction of false positives but is still large enough to avoid false negatives. Note that the challenge of identifying a point location in degenerate polygons is studied by [Schirra 2008] and inaccurate treatment of such degenerate cases is well-known to lead to incorrect collision detection results. By carefully observing the geometric meaning of degeneracy in CCD and understanding where the error comes from, we came up with a novel CCD method that handles degeneracy comprehensively with a tight bound on error tolerance.

Our contribution in this paper is two-fold. Firstly, we introduce a novel area inclusion condition which allows us to obtain a comprehensive error bound. This new condition allows us to analyze how the error in finding coplanarity time can affect the error in flagging positive for the contact at the approximate coplanarity time. Based on this analysis our method avoids *false negatives*, which would be difficult when the barycentric coordinates approach is used. The critical achievement of our method is to incorporate a consistent error tolerance which users can control based on the needs of the application while guaranteeing not to produce any *false negatives*. Secondly, for both algebraic and geometric degeneracy, we present dimensional reduction techniques for degenerate cases. This novel approach simplifies the solution space for the degenerate cases and provides a method to solve such challenging cases in the CCD problem with minimal computation. Our method is clearly demonstrated to provide the robust and accurate results that the *exactCCD* method offers, but it also provides the contact time and maintains the speed of non-exact CCD methods [Hubbard 1994]. Our work excludes the effect of rounding errors, the numerical inaccuracy in square root computations, and the error introduced from file input and output. These are beyond the scope of this paper.

2 Related Work

There are a number of related research projects on collision detection in the graphics community and it is not possible to list all the relevant research in this paper. To be more concise and focused, we shall list only the most recent work in CCD, which is relevant to our current techniques.

In [Lin and Manocha 1993; Mirtich 1997b], the narrow phase collisions between polyhedra are computed using the closest feature points using time coherency. The first time of contact with a threshold using local advancement is used for detecting collisions [Tang et al. 2010a]. However, as simulations get more realistic, exact collision detection between primitive geometry pairs like face-point or edge-edge are more important. Hence in this paper we focus on narrow phase collisions between primitive geometry pairs.

The cubic-solver approach, which was introduced to the field by [Provot 1997], is a *constructive* geometric approach [Brochu et al. 2012] and it has been widely used by the CCD community. To compute the actual time of contact between primitive geometry pairs, [Provot 1997] suggested a method using coplanarity and inclusion tests using barycentric coordinates. In Provot's method, a cubic polynomial equation is formed from the coplanarity condition. Once the time of coplanarity is found by solving this equation, the actual position of each point is computed using the found coplanarity time. Then these positions are used to determine if two geometric entities are actually in collision at that coplanarity time. In such calculations, error analysis is essential to obtain provably correct algorithms, but so far has been missing [Brochu et al. 2012]. The method has in fact been improved on by many researchers (see [Bridson et al. 2002] and references cited therein). However, until now, this approach has not had any control over the error bound. It is also non-deterministic in degenerate cases. The main purpose of our paper is to overcome these well-known shortcomings and our method is the first cubic solver-based method that presents remedies. Provably, the proposed method does not produce any *false negatives*.

Our method is not an *exact CCD* method, since it has a notion of error tolerance. However, numerical experiments demonstrate that, as theory predicts, the proposed method does not produce any *false negatives* and the collision test results are coincident with that of *exactCCD* [Brochu and Bridson 2009; Brochu et al. 2012] for the odd root count collision cases. Note that *exactCCD* can only detect collisions with an odd number of roots to the cubic temporal equation.

3 Notation and Problem Description

In this section we introduce notation following [Yap 2004] and then describe the problems of interest. We define convex hulls from geometric primitives, i.e. faces, edges, and points, and interpret a collision as an intersection between two convex hulls.

Given a point $\mathbf{p} \in \mathbb{R}^3$, whose coordinates are $(p_x, p_y, p_z) \in \mathbb{R}^3$, we shall denote it by

$$\mathbf{p} = \mathsf{Point}(p_x, p_y, p_z). \tag{1}$$

We shall also denote by $\mathsf{CHull}(R)$ a convex hull obtained by a set of points $R = \{\mathbf{p}_i\}_{i=1}^n$, R in $\mathbb{R}^3$ and denote by $|R|$, the *cardinality* of the set R. For notational convenience, whenever we write a bold-faced lower-case letter, it will be a point in $\mathbb{R}^3$.

Definition 3.1 (Linear Movement, the Point) *We consider a one parameter family of zero-dimensional convex hulls or a point moving in time:*

$$\phi : t \in [0,1] \mapsto \mathit{CHull}(\mathbf{p}^t). \tag{2}$$

We say that the point moves linearly in time whenever it has a fixed velocity, namely, it can be written as follows :

$$\phi(t) = \mathbf{p}^t = \mathbf{p}^0 + t\mathbf{v_p} \tag{3}$$

with $\mathbf{p^0}$ and $\mathbf{v_p}$ fixed $\in \mathbb{R}^3$.

Definition 3.2 (Linear Movement, General Convex Hull) *We consider one parameter family of a general convex hulls moving in time given as follows :*

$$\phi : t \in [0,1] \mapsto \mathit{CHull}(R^t), \tag{4}$$

where $R^t = (\mathbf{p}_1^t, \cdots, \mathbf{p}_n^t)$. Note that the mapping ϕ describes the movement of $\mathit{CHull}(R^t)$ in time in the three dimensional Euclidean space $\mathbb{R}^3$. The superscript given for R indicates that

$R^t = (\mathbf{p}_1^t, \cdots, \mathbf{p}_n^t)$ *is a set of points, each of which is dependent on time. We say that the convex hull moves linearly in time whenever each point* $\mathsf{CHull}(\mathbf{p}_i^t)$ *moves linearly in time for all* $i = 1, \ldots, n$.

Definition 3.3 (Contact of Two Convex Hulls) *Given two sets* R_A *and* R_B *of points, we consider two corresponding linearly moving convex hulls,* $\mathsf{CHull}(R_A)$ *and* $\mathsf{CHull}(R_B)$. *We say that two convex hulls are in contact at time* t *iff there exists a point* $\mathbf{p}^t$ *which is included in both convex hulls at time* t.

From this definition, we indicate that a collision of two convex hulls is interpreted as an intersection between two convex hulls. Given this definition of a contact, we shall investigate the following two problems :

Question 3.1 (Face-Point) *Given two sets* R_A^t *and* R_B^t *of points such that* $|R_A^0| = 3$ *and* $|R_B^0| = 1$, *we consider two corresponding linearly moving convex hulls,*

$$\mathsf{CHull}(R_A^t) \quad \textit{and} \quad \mathsf{CHull}(R_B^t),$$

in time interval $[0, 1]$. *We wish to find the earliest time* t *of contact if it exists.*

Question 3.2 (Edge-Edge) *Given two sets* R_A *and* R_B *of points such that* $|R_A^0| = |R_B^0| = 1$, *we consider two corresponding linearly moving convex hulls,*

$$\mathsf{CHull}(R_A^t) \quad \textit{and} \quad \mathsf{CHull}(R_B^t),$$

in time interval $[0, 1]$. *We wish to find the earliest time* t *of contact if it exists.*

Throughout this paper we shall use the standard notation $\times$ for the cross product and $\circ$ for the dot product. The standard Euclidean norm will be denoted by $\|\cdot\|$.

We present a fast and robust narrow-phase continuous collision detection (CCD) method between a linearly moving face and point or between two linearly moving edges within an arbitrarily small user-defined error tolerance E_{tol}.

4 Detection of Contact between a Linearly Moving Face and a Point

We consider a linearly moving face and point from the time $t = 0$ to the time $t = 1$. A face, the convex hull of a set of three points $R^t = \{\mathbf{a}^t, \mathbf{b}^t, \mathbf{c}^t\}$, is denoted by $\mathsf{CHull}(R^t)$. At time t, the face and the point are represented as $\mathsf{CHull}(R^t)$ and a point $\mathbf{p}^t$. At the time of contact $t = \bar{t}$, a face and a point satisfy the following two properties (which are equivalent when a face and a point are in contact).

F-P 1: Four corner points $\{\mathbf{a}^{\bar{t}}, \mathbf{b}^{\bar{t}}, \mathbf{c}^{\bar{t}}, \mathbf{p}^{\bar{t}}\}$ are coplanar.

F-P 2: The point $\mathbf{p}^{\bar{t}}$ belongs to $\mathsf{CHull}(\mathbf{a}^{\bar{t}}, \mathbf{b}^{\bar{t}}, \mathbf{c}^{\bar{t}})$.

The coplanarity in **F-P 1** can be represented as the following mathematical equation:

$$\mathsf{P}(t) = \left\{(\mathbf{a}^t - \mathbf{p}^t) \times (\mathbf{b}^t - \mathbf{p}^t)\right\} \circ (\mathbf{c}^t - \mathbf{p}^t). \tag{5}$$

Eq. (5) is a cubic equation that can be solved analytically or numerically to find collision times $\bar{t}$ where $P(\bar{t}) = 0$. At this step, an approximate root $\widetilde{t} = \bar{t} + \epsilon$ can generally be estimated which is close to the exact collision time plus some numerical error ϵ introduced by the use of iterative root-finding methods [Wilkinson 1963]. Once the approximate coplanarity time $\widetilde{t}$ is known, we investigate whether there is really a contact by checking the condition **F-P 2**. Typically this inclusion test is done by computing the barycentric coordinate of $\mathbf{p}^{\widetilde{t}}$ within $\mathsf{CHull}(\mathbf{a}^{\widetilde{t}}, \mathbf{b}^{\widetilde{t}}, \mathbf{c}^{\widetilde{t}})$. It is well-known that careful attention must be given to this stage since problems such as *false negatives* have been reported when using an inappropriate error tolerance [Brochu and Bridson 2009]. Note that it is this step where an additional tolerance must be assigned, which makes the CCD dependent on multiple and potentially inconsistent tolerances.

4.1 Inclusion Test based on Area Conditions and Consistent Error Tolerance

Typically, the coplanarity time is computed only approximately. This causes problems for the cubic solver approach, because the inclusion test is performed with an approximate coplanarity time $\widetilde{t}$. Computing barycentric coordinates given the approximate coplanarity time $\widetilde{t}$, along with handling the degeneracy in the coplanarity relations, makes the comprehensive error analysis quite challenging, thereby unavoidably introducing *false positives* or *false negatives*.

We shall demonstrate how to avoid the *false negative* in the following discussion. In our method, we introduce the area inclusion condition and provide the error analysis. We note that the area-sum test for the intersection between point-in-polygon is well-known [Chen and Townsend 1987] and dates back to [Nordbeck and Rystedt 1967]. However, their appropriate use in CCD applications do not seem to be available in the literature. In particular, the error analysis provided in this section indicates that once the contact happens and the approximate coplanarity time $\widetilde{t}$ is given within E_{tol} from the exact coplanarity time $\bar{t}$, the contact must be reported if the new condition is met under the consistent error tolerance, i.e., E_{tol} times some initial local mesh size and the size of velocity differences of the face and points. This condition is given in a way that all the possible cases are included exhaustively regardless of the degeneracy arising from collinearity. Provably, the proposed method does not report any *false negatives* for any given error tolerance initially given for the cubic solver, unlike the classical barycentric coordinate approaches. The method can be said to be arbitrarily accurate in finding and detecting the contact as well as the contact time if one can solve the coplanarity time accurately. More importantly, our method does not require multiple tolerances. Rather, a single tolerance for the cubic equation solver can be given, which is consistently used to handle the inclusion test.

4.1.1 Approximate coplanarity time

Numerically one can find the solution to Eq. (5) in many different ways. Appropriate root finding methods include iterative methods such as the bisection or Householder methods [Householder 1970]. Generally, numerical solutions to the cubic equation $\widetilde{t}$ are accurate to within an error tolerance E_{tol} [Wilkinson 1963; Zucker 2008]. Our focus is to show how this error tolerance E_{tol} can be used consistently to guarantee not to produce *false negatives*. Once E_{tol} is set, we can deduce the upper bound of the possible error in the inclusion test (see Appendix A). To find an approximate coplanar time $\widetilde{t}$, we tested numerous root finding methods in our experiments. Householder iteration provided the best performance, and this is method used exclusively in the benchmarks reported below.

4.1.2 Area Inclusion Test

For the inclusion test, we introduce a new and different condition from what is used for the barycentric coordinate approach. Namely, the condition **F-P 2**, shown conceptually in Figure 2, can be checked by the following area condition:

F-P $\widehat{2}$: At the time of contact, $\bar{t}$, the sum of the areas of $\mathsf{CHull}(\mathbf{a}^{\bar{t}}, \mathbf{b}^{\bar{t}}, \mathbf{p}^{\bar{t}})$, $\mathsf{CHull}(\mathbf{a}^{\bar{t}}, \mathbf{c}^{\bar{t}}, \mathbf{p}^{\bar{t}})$ and $\mathsf{CHull}(\mathbf{b}^{\bar{t}}, \mathbf{c}^{\bar{t}}, \mathbf{p}^{\bar{t}})$ is equal to that of $\mathsf{CHull}(\mathbf{a}^{\bar{t}}, \mathbf{b}^{\bar{t}}, \mathbf{c}^{\bar{t}})$.

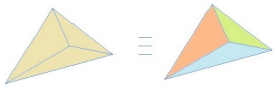

Figure 2: *Two equivalent sets of areas in the vertex and face inclusion test. In contact, the area of a face is equivalent to the sum of the areas of three faces made by the contact point with respect to each edge.*

To simplify our notation, we shall denote the area of the convex hull obtained from three points $\mathsf{CHull}(\mathbf{x}, \mathbf{y}, \mathbf{z})$ by $\mathsf{Area}(\mathsf{CHull}(\mathbf{x}, \mathbf{y}, \mathbf{z}))$. Our area condition and its validity are based on the following Lemma:

Lemma 4.1 *Let $\widetilde{t}$ and $\bar{t}$ be the approximate and exact coplanarity times, respectively, with $|\widetilde{t} - \bar{t}| \leq E_{\text{tol}}$. Let $\mathbf{v}_{\mathbf{x}}$ be the velocity of the linearly moving point $\mathbf{x}^t$. Then the following holds true:*

$$\mathit{Area}(\mathit{CHull}(\mathbf{x}^{\widetilde{t}}, \mathbf{y}^{\widetilde{t}}, \mathbf{z}^{\widetilde{t}})) \leq \mathit{Area}(\mathit{CHull}(\mathbf{x}^{\bar{t}}, \mathbf{y}^{\bar{t}}, \mathbf{z}^{\bar{t}})) + \mathit{E}(\mathbf{x}, \mathbf{y}, \mathbf{z}),$$

where $\mathbf{x}, \mathbf{y}, \mathbf{z}$ can be any $\mathbf{a}, \mathbf{b}, \mathbf{c}$ or $\mathbf{p}$ and

$$\mathit{E}(\mathbf{x}, \mathbf{y}, \mathbf{z}) = E_{\text{tol}} C_1 C_2 + \frac{1}{2} E_{\text{tol}}^2 C_2^2, \tag{6}$$

with $\|\cdot\|$ being the standard Euclidean norm in $\mathbb{R}^3$, and

$$\begin{aligned} C_1 &= \max_{\alpha \neq \beta \in \{\mathbf{x}^0, \mathbf{y}^0, \mathbf{z}^0\}} \|\alpha - \beta\| + \max_{t \in [0,1]} \max_{\alpha \neq \beta \in \{\mathbf{x}, \mathbf{y}, \mathbf{z}\}} t \|\mathbf{v}_\alpha - \mathbf{v}_\beta\|, \\ C_2 &= \max_{\alpha \neq \beta \in \{\mathbf{x}, \mathbf{y}, \mathbf{z}\}} \|\mathbf{v}_\alpha - \mathbf{v}_\beta\|. \end{aligned}$$

The consequence of Lemma 4.1 provides the theoretical error bound for the contact case.

$$\begin{aligned} &\Big|\mathsf{Area}(\mathsf{CHull}(\mathbf{a}^{\widetilde{t}}, \mathbf{b}^{\widetilde{t}}, \mathbf{p}^{\widetilde{t}})) + \mathsf{Area}(\mathsf{CHull}(\mathbf{b}^{\widetilde{t}}, \mathbf{c}^{\widetilde{t}}, \mathbf{p}^{\widetilde{t}})) \\ &\quad + \mathsf{Area}(\mathsf{CHull}(\mathbf{c}^{\widetilde{t}}, \mathbf{a}^{\widetilde{t}}, \mathbf{p}^{\widetilde{t}})) - \mathsf{Area}(\mathsf{CHull}(\mathbf{a}^{\widetilde{t}}, \mathbf{b}^{\widetilde{t}}, \mathbf{c}^{\widetilde{t}}))\Big| \\ &\quad \leq \mathsf{E}(\mathbf{a}, \mathbf{b}, \mathbf{p}) + \mathsf{E}(\mathbf{b}, \mathbf{c}, \mathbf{p}) + \mathsf{E}(\mathbf{c}, \mathbf{a}, \mathbf{p}) + \mathsf{E}(\mathbf{a}, \mathbf{b}, \mathbf{c}) \end{aligned} \tag{7}$$

which is because at the time of contact $t = \bar{t}$, the following identity holds true:

$$\begin{aligned} &\mathsf{Area}(\mathsf{CHull}(\mathbf{a}^{\bar{t}}, \mathbf{b}^{\bar{t}}, \mathbf{c}^{\bar{t}})) = \mathsf{Area}(\mathsf{CHull}(\mathbf{a}^{\bar{t}}, \mathbf{p}^{\bar{t}}, \mathbf{c}^{\bar{t}})) \\ &\quad + \mathsf{Area}(\mathsf{CHull}(\mathbf{a}^{\bar{t}}, \mathbf{b}^{\bar{t}}, \mathbf{p}^{\bar{t}})) + \mathsf{Area}(\mathsf{CHull}(\mathbf{b}^{\bar{t}}, \mathbf{c}^{\bar{t}}, \mathbf{p}^{\bar{t}})). \end{aligned}$$

The area condition indicates that if there is contact, then the condition (7) must be satisfied. In other words, it is a sufficient condition to detect collisions. We remark that the aforementioned bound is rigorous enough in the sense that unlike the classical barycentric coordinate cases it does not produce any *false negatives*. It is also more consistent, since our methodology can employ only a single tolerance, which is needed only for the cubic solver. Note that C_1 and C_2 pertain to the size of velocity differences like $\|\mathbf{v}_\mathbf{a} - \mathbf{v}_\mathbf{p}\|$ and initial edge lengths like $\|\mathbf{c}^0 - \mathbf{a}^0\|$ and they are all computable.

While condition (7) does not allow any *false negatives*, it may report *false positives* within the given tolerance. Such *false positives* are the cases where a point and face are close enough to satisfy the condition (7) but not in geometrically exact contact. However, in our method the error tolerance can be arbitrarily small. Hence users can adjust the error tolerance for their application needs.

4.2 Treatment of Algebraically Degenerate Cases

In solving Eq. (5), we can encounter algebraically degenerate cases when all four coefficients of the polynomial are zero. We handle such cases by considering the geometric meaning of the polynomial coefficients. Without correctly handling these cases, we could miss collisions. By rearranging Eq. (5), we have

$$\begin{aligned} \boldsymbol{F}(t) \;=\; & [(\boldsymbol{v} \times \boldsymbol{u}) \circ \boldsymbol{w}]\, t^3 \\ & + [(\boldsymbol{\alpha} \times \boldsymbol{u} + \boldsymbol{v} \times \boldsymbol{\beta}) \circ \boldsymbol{w} + (\boldsymbol{v} \times \boldsymbol{u}) \circ \boldsymbol{\gamma}]\, t^2 \\ & + [(\boldsymbol{\alpha} \times \boldsymbol{u} + \boldsymbol{v} \times \boldsymbol{\beta}) \circ \boldsymbol{\gamma} + (\boldsymbol{\alpha} \times \boldsymbol{\beta}) \circ \boldsymbol{w}]\, t \\ & + (\boldsymbol{\alpha} \times \boldsymbol{\beta}) \circ \boldsymbol{\gamma} = \mathbf{0}, \end{aligned} \tag{8}$$

where

$$\begin{aligned} \boldsymbol{\alpha} &= \mathbf{a}^0 - \mathbf{p}^0, \\ \boldsymbol{\beta} &= \mathbf{b}^0 - \mathbf{p}^0, \\ \boldsymbol{\gamma} &= \mathbf{c}^0 - \mathbf{p}^0, \\ \boldsymbol{v} &= \mathbf{a}^1 - \mathbf{a}^0 - (\mathbf{p}^1 - \mathbf{p}^0), \\ \boldsymbol{u} &= \mathbf{b}^1 - \mathbf{b}^0 - (\mathbf{p}^1 - \mathbf{p}^0), \\ \boldsymbol{w} &= \mathbf{c}^1 - \mathbf{c}^0 - (\mathbf{p}^1 - \mathbf{p}^0). \end{aligned}$$

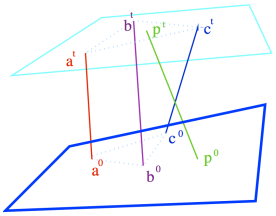

Figure 3: *Initially $\mathbf{a}^0, \mathbf{b}^0, \mathbf{c}^0$ and $\mathbf{p}^0$ are coplanar, but $\mathbf{p}^0$ is not in the convex hull generated by the other three points. At the time t, it is now in the convex hull of three points $\mathbf{a}^t, \mathbf{b}^t$ and $\mathbf{p}^t$*

Let all four coefficients of Eq. (5) be zero. Then $P(t) = 0$ regardless of time t. In other words, the four points remain coplanar throughout the interval $[0, 1]$. From this observation we know that only the following two cases can lead to a contact:

Case 1: The face and the point are already in contact at time $t = 0$.

Case 2: The face and the point are not in contact initially but the point is in contact with at least one edge of the face during (0,1].

Therefore we now know that in such cases the problem can be reduced to a collision between a line segment and a point.

4.2.1 Dimensional Reduction Techniques

For the algebraically degenerate cases, we are able to reduce the CCD problem to solving three quadratic equations. Without loss of generality, we shall assume $\mathbf{p}^t$ is colliding with the edge connecting $\mathbf{a}^t$ and $\mathbf{b}^t$, which results in the following equation:

$$(\mathbf{a}^t - \mathbf{p}^t) \times (\mathbf{b}^t - \mathbf{p}^t) = \mathbf{0}. \tag{9}$$

Eq. (9) is a quadratic equation given by

$$(\boldsymbol{v} \times \boldsymbol{u})t^2 + (\boldsymbol{\alpha} \times \boldsymbol{u} + \boldsymbol{v} \times \boldsymbol{\beta})t + \boldsymbol{\alpha} \times \boldsymbol{\beta} = \mathbf{0}, \tag{10}$$

where

$$\begin{aligned} \boldsymbol{\alpha} &= \mathbf{a}^0 - \mathbf{p}^0, \quad \boldsymbol{\beta} = \mathbf{b}^0 - \mathbf{p}^0, \\ \boldsymbol{v} &= \mathbf{a}^1 - \mathbf{a}^0 - \mathbf{p}^1 + \mathbf{p}^0, \\ \boldsymbol{u} &= \mathbf{b}^1 - \mathbf{b}^0 - \mathbf{p}^1 + \mathbf{p}^0. \end{aligned}$$

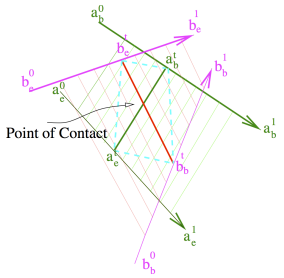

Figure 4: *Two linearly moving edges meet at the point of contact.*

Two similar equations from the other edges can be derived as well. Roots of these three equations have to be checked in order to decide the earliest detection times for all three edges.

We would like to remark that another algebraic degeneracy can occur for Eq. (10): when all three coefficients are zero, all three points are collinear throughout the interval. In such cases, to have a contact it is the case that either 1) the three points are already in contact, or 2) the point meets the end point of an edge within the interval. Here we reduce our problem domain one step further. Note that since we are interested in the first contact, we have to pick the earliest root among identified roots.

5 Detection of Contact between Two Linearly Moving Edges

Similarly to the face-point case, we formulate our methodology for edge-edge contact detection. We consider two line segments or edges that are moving linearly from time $t = 0$ to 1. The two moving edges are $\mathsf{CHull}(R_A^t)$ with $R_A^t = \{\mathbf{a}_b^t, \mathbf{a}_e^t\}$ and $\mathsf{CHull}(R_b^t)$ with $R_B^t = \{\mathbf{b}_b^t, \mathbf{b}_e^t\}$. Note that again, the argument t denotes the time belonging to the interval $[0, 1]$ and each point is in $\mathbb{R}^3$. For example, $\mathbf{a}_b^t = ((a_b^t)_x, (a_b^t)_y, (a_b^t)_z)^T \in \mathbb{R}^3$.

Given any two points $\mathbf{p}^0$ and $\mathbf{p}^1$, we consider the line connecting them and starting at $\mathbf{p}^0$:

$$\mathbf{p}^t = \mathbf{p}^0 + t(\mathbf{p}^1 - \mathbf{p}^0). \tag{11}$$

We have the following relations:

$$\begin{aligned}
\mathbf{a}_b^t &= \mathbf{a}_b^0 + t(\mathbf{a}_b^1 - \mathbf{a}_b^0),\\
\mathbf{a}_e^t &= \mathbf{a}_e^0 + t(\mathbf{a}_e^1 - \mathbf{a}_e^0),\\
\mathbf{b}_b^t &= \mathbf{b}_b^0 + t(\mathbf{b}_b^1 - \mathbf{b}_b^0),\\
\mathbf{b}_e^t &= \mathbf{b}_e^0 + t(\mathbf{b}_e^1 - \mathbf{b}_e^0).
\end{aligned}$$

Figure (4) shows two linearly moving edges which meet at a point at time t. We would like to remark that at the time t of contact, the following two conditions are met.

E-E 1: Four corner points $\{\mathbf{a}_b^t, \mathbf{a}_e^t, \mathbf{b}_b^t, \mathbf{b}_e^t\}$ are coplanar.

E-E 2: The point of contact belongs to $\mathsf{CHull}(\mathbf{a}_b^t, \mathbf{a}_e^t, \mathbf{b}_b^t, \mathbf{b}_e^t)$.

We note that the condition **E-E 1** can be written as

$$\left\{(\mathbf{a}_e^t - \mathbf{a}_b^t) \times (\mathbf{b}_b^t - \mathbf{a}_b^t)\right\} \circ (\mathbf{b}_e^t - \mathbf{a}_b^t) = 0. \tag{12}$$

Similar to the face-point case, to use a consistent error tolerance we introduce the alternative condition **E-E $\widehat{2}$**:

E-E $\widehat{2}$: The sum of areas for $\mathsf{CHull}(\mathbf{a}_b^t, \mathbf{a}_e^t, \mathbf{b}_b^t)$ and $\mathsf{CHull}(\mathbf{a}_b^t, \mathbf{a}_e^t, \mathbf{b}_e^t)$ is equal to the sum of areas for $\mathsf{CHull}(\mathbf{b}_b^t, \mathbf{b}_e^t, \mathbf{a}_b^t)$ and $\mathsf{CHull}(\mathbf{b}_b^t, \mathbf{b}_e^t, \mathbf{a}_e^t)$.

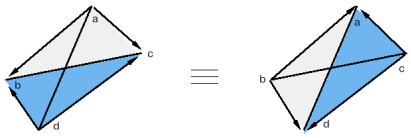

Figure 5: *In contact, the sum of areas by* $\mathsf{CHull}(\mathbf{a}, \mathbf{b}, \mathbf{c})$ *and* $\mathsf{CHull}(\mathbf{b}, \mathbf{d}, \mathbf{c})$ *is equal to that of* $\mathsf{CHull}(\mathbf{a}, \mathbf{d}, \mathbf{c})$ *and* $\mathsf{CHull}(\mathbf{a}, \mathbf{b}, \mathbf{d})$.

This is shown schematically in Figure (5). Similar to the face-point case, degenerate cases can be handled based on the polynomial coefficients using dimensional reduction. Only a small change is required in dimensional reduction since collinearity has to be checked for each edge.

6 Results

In this section we compare the results of vertex-face collision detection obtained by our method with the other CCD methods such as *exactCCD* [Brochu et al. 2012], *safeCCD* [Wang 2014] and *openCCD* [Kim and Yoon 2009]. For consistency, each method is compared using the same platform (hardware and operating system) that was used for the original implementation. Benchmarks are performed with random data and also with dynamic data. The dynamic data used in our benchmarks is from [Tang et al. 2009; Yoon et al. 2007; Tang et al. 2012]. For random data, each position at time $t = 0$ and 1 is generated within the unit cube.

First, we compare our method to *exactCCD* and *safeCCD* with random data. Typically the cubic equation we get from random data is well shaped. To implement the cubic solver, we first identify the local minima and maxima on the interval $[0, 1]$ using the derivatives of the cubic equation. We then check whether $f(t)$ changes sign on each interval, and if the sign changes then we know a root exists and we invoke the Householder routine to find its value.

Given an interval, one can have up to three contacts, since a coplanarity equation can have up to three roots. However, *exactCCD* only reports a collision for cases with an odd number of roots. Hence, we modify our method to collect the even and odd root count collisions separately. This alteration does not affect the performance since only the interpretation changes for a given collision result. For 10 million random data collision pairs, our method took about 7.8–10 sec while *exactCCD* took 37.4 sec to produce the exact same results for odd collisions. For *safeCCD* it was 918,663 collisions in 13.6 sec. The coefficient B in *safeCCD* is computed for each collision pair while the relative velocities are used to get the maximum of velocity magnitudes.

This is shown in Figure 1 and Table 1. Our method is not only faster but also provides more details like the time of contact and it reports collisions corresponding to an even number of roots. The comparison was using performed using Mac OS X (Version 10.9.2) running on a 3.2 GHz Intel Core i5 processor with 8GB 1600 MHz DDR3 memory and Clang LLVM C++ compiler. All floating point data is stored in double precision.

Table 2 and Figure 7 show the comparison of results using dynamic data. The narrow-phase collision detection candidates among face-vertex pairs have been culled using an axis-aligned bounding box test. The broad-phase culling results are pre-computed and stored so they do not affect the benchmark timing results. Only the pre-computed narrow-phase candidates are tested in the benchmarks. In *fastCCD*, we used an error tolerance of $E_{\text{tol}} = 10^{-9}$.

Our benchmarks show that *fastCCD* consistently outperformed *exactCCD* and *safeCCD* in terms of computational efficiency and ac-

curacy in all tests using UNC dynamic data. We observed that the performance advantage of *fastCCD* over *exactCCD* is proportional to the number of degenerate cases. Note that at times *exactCCD* and *fastCCD* report different numbers of collisions. For those cases we have investigated so far, these different results are the cases of near collisions but not actual collisions (*false positives*). Since the error tolerance in *exactCCD* is not defined , it is unknown how these near collisions are identified as collisions in *exactCCD*.

We compare our method with *openCCD* using random data collision pairs. For this comparison, we modify *openCCD* to expose and directly call the internal `Intersect_VF` function. This avoids unnecessary overhead in traversing their internal tree structure. As shown in Figure 8, *fastCCD* is about 4.5-5.6 times faster than *openCCD*. Note that in this test, all the even root count collisions are counted as well. The comparison with *openCCD* has been performed on Windows 8.1 running on an Intel Core i7-4770 CPU @3.40GHz with 8GB RAM and a 64-bit operating system. *fastCCD* is implemented using Visual Studio 2013 in **double** precision, while *openCCD* is compiled in Visual Studio 2008 in **float** precision. We do not know how much this affects the performance comparison. Both methods are compiled with Release-Win32 options.

7 Discussion

In this paper, we resolve the main issues in the cubic-solver-based narrow-phase CCD method by providing a consistent error tolerance and comprehensively handling geometric degeneracy. Our method is provably free of *false negatives* when using a consistent error tolerance. Benchmarking shows that our method is faster than other known methods, while providing detailed collision information like the time of contact and even root count for collisions. In this paper, we assumed a linear trajectory for primitives. Curved trajectories will be an interesting area of future research, especially for applications with large time intervals.

A Proof of Lemma 4.1

In this section we provide the proof of the Lemma 4.1. Let $\tilde{t}$ and $\bar{t}$ be the solution to the cubic equation obtained with tolerance E_{tol} and the exact coplanarity time, respectively. We note that for $\mathbf{x}$ being $\mathbf{a}, \mathbf{b}, \mathbf{c}$ or $\mathbf{p}$, since

$$\mathbf{x}^{\bar{t}} = \mathbf{x}^0 + \bar{t}\mathbf{v}_\mathbf{x} \quad \text{and} \quad \mathbf{x}^{\tilde{t}} = \mathbf{x}^0 + \tilde{t}\mathbf{v}_\mathbf{x}$$

the following inequality holds true :

$$\begin{aligned}
&2\text{Area}(\text{CHull}(\mathbf{x}^{\tilde{t}}, \mathbf{y}^{\tilde{t}}, \mathbf{z}^{\tilde{t}})) = |(\mathbf{y}^{\tilde{t}} - \mathbf{x}^{\tilde{t}}) \times (\mathbf{z}^{\tilde{t}} - \mathbf{x}^{\tilde{t}})| \\
&= |[\mathbf{y}^{\bar{t}} - \mathbf{x}^{\bar{t}} + (\mathbf{y}^{\tilde{t}} - \mathbf{y}^{\bar{t}}) - (\mathbf{x}^{\tilde{t}} - \mathbf{x}^{\bar{t}})] \\
&\quad \times [\mathbf{z}^{\bar{t}} - \mathbf{x}^{\bar{t}} + (\mathbf{z}^{\tilde{t}} - \mathbf{z}^{\bar{t}}) - (\mathbf{x}^{\tilde{t}} - \mathbf{x}^{\bar{t}})]| \\
&\leq 2\text{Area}(\text{CHull}(\mathbf{x}^{\bar{t}}, \mathbf{y}^{\bar{t}}, \mathbf{z}^{\bar{t}})) \qquad (13) \\
&\quad + |(\mathbf{y}^{\bar{t}} - \mathbf{x}^{\bar{t}}) \times [(\mathbf{z}^{\tilde{t}} - \mathbf{z}^{\bar{t}}) - (\mathbf{x}^{\tilde{t}} - \mathbf{x}^{\bar{t}})]| \\
&\quad + |[(\mathbf{y}^{\tilde{t}} - \mathbf{y}^{\bar{t}}) - (\mathbf{x}^{\tilde{t}} - \mathbf{x}^{\bar{t}})] \times (\mathbf{z}^{\bar{t}} - \mathbf{x}^{\bar{t}})| \\
&\quad + |[(\mathbf{y}^{\tilde{t}} - \mathbf{y}^{\bar{t}}) - (\mathbf{x}^{\tilde{t}} - \mathbf{x}^{\bar{t}})] \times [(\mathbf{z}^{\tilde{t}} - \mathbf{z}^{\bar{t}}) - (\mathbf{x}^{\tilde{t}} - \mathbf{x}^{\bar{t}})]|.
\end{aligned}$$

The last three terms in the aforementioned inequality can be bound by the following expression:

$$\begin{aligned}
&E_{\text{tol}}(\|\mathbf{y}^0 - \mathbf{x}^0\| + t\|\mathbf{v}_\mathbf{y} - \mathbf{v}_\mathbf{x}\|)(\|\mathbf{v}_\mathbf{z} - \mathbf{v}_\mathbf{x}\|) \\
&\quad + E_{\text{tol}}(\|\mathbf{z}^0 - \mathbf{x}^0\| + t\|\mathbf{v}_\mathbf{z} - \mathbf{v}_\mathbf{x}\|)(\|\mathbf{v}_\mathbf{y} - \mathbf{v}_\mathbf{x}\|) \quad (14) \\
&\quad + E_{\text{tol}}^2\|\mathbf{v}_\mathbf{y} - \mathbf{v}_\mathbf{x}\|\|\mathbf{v}_\mathbf{z} - \mathbf{v}_\mathbf{x}\|.
\end{aligned}$$

This completes the proof of Lemma 4.1.

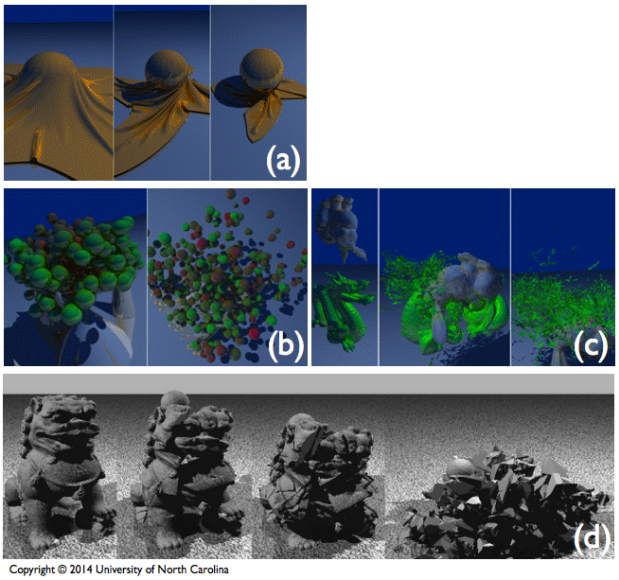

Figure 6: *Benchmark scene visualization provided by [Tang et al. 2012]. (a) cloth-ball simulation model, (b) N-body simulation, (c) exploding dragon and bunny model and (d) lion.*

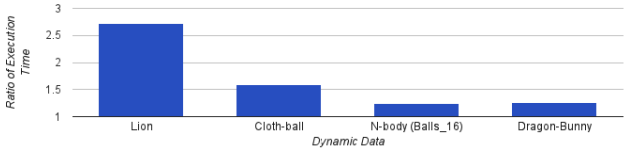

Figure 7: *Ratio of execution time for fastCCD to openCCD when processing collisions from the UNC dynamic scene data.*

Table 2: *Detailed benchmark results for the UNC dynamic scene data.* Degenerate *cases occur when all coefficients of a cubic equation are zero.* ZeroA *cases occur when the coefficient of t^3 is zero, meaning any two of four velocities are parallel.*

		Lion	Cloth-ball	N-Body(Balls_16)	Dragon-Bunny
# of frames		44	93	75	15
Total Pairs		227,217,729	31,170,312	427,022,018	1,343,153,663
Degenerate Cases		0.001%	11%	0%	0 %
ZeroA Cases		5%	0.3%	79 %	0.5%
fastCCD	Num of Collisions	1,734,172	65,182	288,694	2,836,492
	Time(sec)	35.6	6.4	64.8	280.3
	Error Tol.	1×10^{-9}	1×10^{-9}	1×10^{-9}	1×10^{-9}
exactCCD	Num of Collisions	1,734,449	65,182	298,671	2,843,949
	Time(sec)	97.3	10.8	81.6	356.45
safeCCD	Num of Collisions	2,434,539	73,279	535,830	5,690,344
	Time(sec)	55.8	10.5	108.6	546.4

Figure 8: *Ratio of execution time for fastCCD to openCCD when processing one million randomly generated collision pairs. With this input data openCCD found 77192 collisions in 8.704 sec. For approximately $E_{tol} = 1 \times 10^{-9}$ or lower, fastCCD produces the same result (shown in the highlighted region). For lower values of the error tolerance, fastCCD reports a small number of additional collisions.*

References

BRIDSON, R., FEDKIW, R., AND J., A. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. (Proc. SIGGRAPH) 21*, 594–603.

BROCHU, T., AND BRIDSON, R. 2009. Numerically robust continuous collision detection for dynamic explicit surfaces. Tech. rep., University of British Columbia.

BROCHU, T., EDWARDS, E., AND BRIDSON, R. 2012. Efficient geometrically exact continuous collision detection. *ACM Trans. Graph. 31*, 4 (July), 96:1–96:7.

CHEN, M., AND TOWNSEND, T. 1987. Efficient and consistent algorithms for determining the containment of points in polygons and polyhedra. In *Eurographics*, G. Marechal, Ed. Elsevier Science Publishers (North-Holland), 423–437.

ERLEBEN, K., SPORRING, J., HENRIKSEN, K., AND DOHLMANNF, H. 2005. Physics-based animation. *Charles River Media 613616*.

HAHN, J. K. 1988. Realistic animation of rigid bodies. *Computer Graphics 22(4) August*.

HARMON, D., VOUGA, E., TAMSTORF, R., AND GRINSPUN, E. 2008. Robust treatment of simultaneous collisions. *AMC Trans. Graph (Proc. SIGGRAPH) 27*, 1–4.

HOUSEHOLDER, A. S. 1970. The numerical treatment of a single nonlinear equation. *McGraw-Hill. p. 169*.

HUBBARD, P. M. 1994. Collision detection for interactive graphics applications. PhD thesis. *Brown University, October 1994*.

KIM, D., AND YOON, S.-E., 2009. OpenCCD: Continuous Collision Detection API.

LIN, M., AND MANOCHA, D. 1993. Interference detection between curved objects for computer animation. *In Models and Techniques in Computer Animation. Springer-Verlag*.

MIRTICH, B. 1997. Efficient algorithms for two-phase collision detection. *Technical Report, Mitsubishi Electric Research Laboratories*.

MIRTICH, B. 1997. V-clip: fast and robust polyhedral collision detection. *Technical Report TR97-05, Mitsubishi Electric Research Lab, Cambridge, MA, July*.

MOORE, M., AND WILHEMS, J. 1988. Collision detection and response for computer animation. *Computer Graphics 22(4) August*.

NORDBECK, S., AND RYSTEDT, B. 1967. Computer Cartography Point-In-Polygon Problems. *BIT 7*, 39–64.

PROVOT, X. 1997. Collision and self-collision handling in cloth model dedicated to design garment. *Graphics Interface 177*.

SCHIRRA, S. 2008. *Algorithms-ESA 2008*. Spring-Verlag, ch. How reliable are practical point-in-polygon strategies?, 744–755.

TANG, M., CURTIS, S., YOON, S.-E., AND MANOCHA, D. 2009. ICCD: Interactive continuous collision detection between deformable models using connectivity-based culling. *IEEE Transactions on Visualization and Computer Graphics 15*, 4, 544–557.

TANG, M., KIM, Y. J., AND MANOCHA, D. 2010. Continuous collision detection for non-rigid contact computations unsing local advancement. *IEEE International Conference on Robotics and Automation Anchorage Convention District*.

TANG, M., MANOCHA, D., AND TONG, R. 2010. Fast continuous collision detection using deforming non-penetration filters. In *I3D '10: Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, ACM, 7–13.

TANG, M., CURTIS, S., YOON, S.-E., AND MANOCHA, D. 2012. UNC dynamic scene benchmarks. Retrieved from UNC Dynamic Scene Benchmarks web site: `http://gamma.cs.unc.edu/DYNAMICB/`.

WANG, H. 2014. Defending continuous collision detection against errors. *ACM Transactions on Graphics (SIGGRAPH) 33*, 4.

WILKINSON, J. 1963. *Rounding Errors in Algebraic Processes*. Prentice Hall.

WITKIN, A., AND BARAFF, D. 1997. An introduction to physically based modeling: Rigid body simulation II : Nonpenetration constratins. *SIGGRAPH Courses*.

WOOP, S., BENTHIN, C., AND WALD, I. 2013. Watertight ray/triangle intersection. *Journal of Computer Graphics Techniques (JCGT) 2*, 1 (June), 65–82.

YAP, C. 2004. *Robust Geometric Computation*, 2nd ed., vol. CRS Handbook of Computational and Discrete Geometry. CRC Press.

YOON, S.-E., CURTIS, S., AND MANOCHA, D. 2007. Ray tracing dynamic scenes using selective restructuring. In *Eurographics Symposium and Rendering*, ACM.

ZUCKER, I. 2008. Real roots of cubics: explicit formula for quasi-solutions. *The Mathematical Gazette 92*, 269–276.