Defending Continuous Collision Detection against Errors

Huamin Wang[*]
The Ohio State University

Abstract

Numerical errors and rounding errors in continuous collision detection (CCD) can easily cause collision detection failures if they are not handled properly. A simple and effective approach is to use error tolerances, as shown in many existing CCD systems. Unfortunately, finding the optimal tolerance values is a difficult problem for users. Larger tolerance values will introduce false positive artifacts, while smaller tolerance values may cause collisions to be undetected. The biggest issue here is that we do not know whether or when CCD will fail, even though failures are extremely rare. In this paper, we demonstrate a set of simple modifications to make a basic CCD implementation failure-proof. Using error analysis, we prove the safety of this method and we formulate suggested tolerance values to reduce false positives. The resulting algorithms are safe, automatic, efficient, and easy to implement.

Keywords: Floating-point arithmetic, rounding error, numerical error, continuous collision detection, quadratic and cubic solver.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics—Animation.

Links: DL PDF

1 Introduction

Detecting the existence of exact vertex-triangle or edge-edge intersection within a time step is the fundamental step in continuous collision detection (CCD). Due to numerical and rounding errors, the detection result may be false in two ways: *false negatives*, when CCD fails to detect real collision events; and *false positives*, when CCD reports collisions while real ones do not happen. Since false negatives have more severe consequences than false positives, they are considered as collision detection failures, which must be strictly avoided. Proposed by Bridson et al. [2002], a typical approach used in existing systems is to report more collision events than necessary by setting error tolerance thresholds. This approach can effectively reduce collision failures, but it is still not clear how large the tolerances must be to avoid failures completely. To play it safe, user typically chooses larger tolerance values, but the extra false positives may trigger other problems, such as visual artifacts, computational burden, and convergence issues in collision handling. To solve this problem, Brochu and Bridson [2012] studied CCD from a new geometric perspective and they formulated a collision predicate that can eliminate both false negatives and false positives. Unfortunately, it is computationally expensive due to more arithmetic operations involved in exact and interval arithmetic, the latter of which is applied to reduce the use of more expensive exact arithmetic.

Figure 1: *After a set of simple modifications, cubic-based CCD algorithms can become failure-proof when simulating the animation of a bow knot, in which complex collisions occur.*

A typical CCD method solves a cubic equation to find the time when two primitives are coplanar and uses it to detect intersection. Since the use of error tolerances has already demonstrated its effectiveness in this method, an important question is: *can it be strictly failure-proof?* Obviously, an algorithm is trivially and uselessly "failure-proof", if it always reports positive. So our real interest is to avoid failures and reduce false positives at the same time. Unfortunately, our research showed that it is difficult to limit the false positives caused by existing algorithms. To handle this issue, our idea is to introduce a number of conditions, under which vertex-triangle and edge-edge tests are guaranteed to be failure-proof. If any condition is not satisfied, we resort to vertex-edge CCD test, which reports a collision if the vertex-edge distance drops below a certain threshold within a time step. By using a sufficiently large distance threshold, we ensure the detection of any collision event.

While our idea is simple, its implementation forces us to face two critical challenges. The first challenge is how to formulate vertex-edge test. Computing the time when the Euclidean vertex-edge distance is equal to a threshold involves solving a quartic equation, which is computationally expensive and susceptible to more errors. The volume-based geometric predictor proposed by Brochu and Bridson [2012] can detect the short distance without computing the time, but it is not robust against rounding errors if not using exact arithmetic. The second challenges is how to evaluate the overall test error. Since the numerical error caused by the cubic solver and the rounding errors caused by floating-point arithmetic coexist, we cannot perform direct error analysis as in [Wilkinson 1994].

In this work, we made the following contributions: 1) a novel implementation of vertex-edge test with $O(\epsilon^{\frac{1}{4}})$ relative error, in which ϵ is machine epsilon; 2) systematic error analysis of the cubic-based tests; 3) a set of simple modifications to make cubic-based tests failure-proof; and 4) the formulation of lower bounds on the error tolerances. Compared with existing methods, our method is:

Safe. We prove that cubic-based CCD algorithms can be unconditionally failure-proof after applying our methods.

[*]Email: whmin@cse.ohio-state.edu

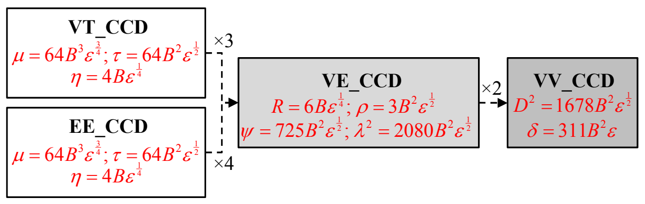

VT_CCD
$\mu = 64B^3\varepsilon^{\frac{3}{4}}; \tau = 64B^2\varepsilon^{\frac{1}{2}}$
$\eta = 4B\varepsilon^{\frac{1}{4}}$
EE_CCD
$\mu = 64B^3\varepsilon^{\frac{3}{4}}; \tau = 64B^2\varepsilon^{\frac{1}{2}}$
$\eta = 4B\varepsilon^{\frac{1}{4}}$
×3
×4
VE_CCD
$R = 6B\varepsilon^{\frac{1}{4}}; \rho = 3B^2\varepsilon^{\frac{1}{2}}$
$\psi = 725B^2\varepsilon^{\frac{1}{2}}; \lambda^2 = 2080B^2\varepsilon^{\frac{1}{2}}$
×2
VV_CCD
$D^2 = 1678B^2\varepsilon^{\frac{1}{2}}$
$\delta = 311B^2\varepsilon$

Figure 2: *Our CCD tests and their error tolerance values. A VT test may call the VE test up to three times, and an EE test may call the VE test up to four times. The choices of tolerance values will be discussed in Section 4. Note that they have been slightly increased in our analysis, so they can be computed using floating-point arithmetic directly.*

Simple. Our modifications are simple and can be easily incorporated into existing systems.

Efficient. Our experiment shows that the computational overhead of our method is only a fraction of the original test cost.

Automatic. Based on error analysis, CCD algorithms can automatically determine all of the tolerance values. User does not need to specify a single parameter.

Accurate. Our method computes the tolerance values by minimizing the vertex-triangle or edge-edge distance when a false positive occurs. So using them can reduce false positives.

2 Related Work

Collision detection. Errors and degeneracies in proximity and intersection tests have been studied [Gilbert et al. 1988; Goldberg 1991; Shewchuk 1996; Larsen et al. 2000] in the past. Based on these tests, researchers developed the discrete collision detection approach [Baraff et al. 2003; Volino and Magnenat-Thalmann 2006] that tests collisions at discrete time instants only. While this approach is robust, it may miss collisions when objects move fast, known as the *tunneling* artifact. One solution is to gradually advance the simulation time until a collision happens, as proposed by Tang et al. [2010]. Alternatively, the continuous collision detection approach computes the possible collision time when two primitives are coplanar. For vertex-triangle and edge-edge collisions, the time can be found by solving cubic equations as Provot [1997] showed. Bridson et al. [2002] improved the robustness of this method by using rounding error tolerances on vertex-plane distances, line-line distances, and barycentric coordinates. However, how to find optimal error tolerances is a tricky problem and there is often no way to guarantee the detection of every collision event. To avoid collision failures, Brochu and Bridson [2012] proposed to separate collision detection from collision time query, and developed a root-parity-based CCD algorithm using exact arithmetic. The robustness of collision tests can also be improved by reporting a collision when the distance between two primitives is less than a threshold as Stam [2009] suggested. This requires solving six-order polynomial equations, when handling vertex-triangle or edge-edge collisions.

Collision culling and handling. Recent research in collision culling has resulted in a number of efficient techniques [Schvartzman et al. 2010; Pabst et al. 2010; Lauterbach et al. 2010; Tang et al. 2011; Zheng and James 2012]. Our method is fully compatible with them. Once collisions are found, the next issue is how to resolve collisions by applying collision responses on primitives. This can be difficult in complex multi-collision cases. A typical solution to unresolved collisions after a number of processing steps is to treat them as rigid bodies in impact zones [Provot 1997; Bridson et al. 2002; Huh et al. 2001; Harmon et al. 2008]. Thomaszewski et al. [2008] suggested to process collisions asynchronously for more accurate collision responses, although their method still relies on synchronous collision handling to avoid remaining collisions. A purely asynchronous system was developed by Harmon et al. [2009; 2011], which guarantees that each collision process can be done in a finite time and its result is free of penetrations. Ainsley et al. [2012] combined an asynchronous system with continuous collision detection, which reduces the number of rescheduling events and makes the system faster. Since the use of our method in CCD reduces false positives and makes the collision time more accurate, we believe it will improve the robustness of collision handling as well.

Algorithm 1 VV_CCD($\mathbf{x}_i^0, \mathbf{x}_i^1, \mathbf{x}_j^0, \mathbf{x}_j^1, D^2$)

```
Compute $\mathbf{x}_{ji}$ and $\mathbf{v}_{ji}$;
$\mathbf{T} \leftarrow \texttt{Clamp}(-(\mathbf{x}_{ji} \cdot \mathbf{v}_{ji})/(\mathbf{v}_{ji} \cdot \mathbf{v}_{ji}), 0, 1)$;
for every root $t \in \mathbf{T}$ do                              ▷ Proximity test
    if $\left\|\mathbf{x}_{ji} + t\mathbf{v}_{ji}\right\|_2^2 \leq D^2 + \delta$ then return true;
```

3 Continuous Collision Detection

In this section, we will present our algorithms for vertex-vertex (VV) CCD test, vertex-edge (VE) CCD test, vertex-triangle (VT) CCD test, and edge-edge (EE) CCD test. Our basic idea is to incorporate the VV test into the VE test, and the VE test into the VT and EE tests, as Figure 2 shows. We note that the pseudo code given in this section slightly differs from the descriptions, due to the use of error tolerances. We will discuss them later in Section 4.

3.1 Vertex-Vertex Collision Detection

Vertex-vertex CCD reports a collision, if the Euclidean distance is shorter than a threshold D at any time t between the starting time 0 and the ending time 1. The Euclidean distance is defined as: $(\mathbf{x}_{ji} + t\mathbf{v}_{ji}) \cdot (\mathbf{x}_{ji} + t\mathbf{v}_{ji})$, in which $\mathbf{x}_{ji} = \mathbf{x}_{ji}^0 = \mathbf{x}_j^0 - \mathbf{x}_i^0$ and $\mathbf{v}_{ji} = (\mathbf{x}_j^1 - \mathbf{x}_i^1) - (\mathbf{x}_j^0 - \mathbf{x}_i^0)$. We compute t_0 when the distance is minimized as: $t_0 = -(\mathbf{x}_{ji} \cdot \mathbf{v}_{ji})/(\mathbf{v}_{ji} \cdot \mathbf{v}_{ji})$. Since t_0 must be in $[0, 1]$, we apply `Clamp` on it. After that, we compute the squared distance at t_0. If it is less than D^2, the algorithm reports a collision as Algorithm 1 shows.

3.2 Vertex-Edge Collision Detection

Vertex-edge CCD reports a collision if the Euclidean distance is less than a threshold R. Let $\mathbf{x}_0$ be a vertex and $\mathbf{x}_i\mathbf{x}_j$ be an edge. The Euclidean distance between the vertex and the edge line at time t is: $\frac{\|(\mathbf{x}_{0i}+t\mathbf{v}_{0i})\times(\mathbf{x}_{ji}+t\mathbf{v}_{ji})\|_2}{\|\mathbf{x}_{ji}+t\mathbf{v}_{ji}\|_2}$. Finding the time when it gets minimized (or equal to R) is difficult without solving a complex equation. But since $\|\mathbf{x}\|_1 \leq \sqrt{3}\|\mathbf{x}\|_2 \leq 3\|\mathbf{x}\|_\infty$ for any $\mathbf{x}$, if the Euclidean distance is less than R at t, then the following condition must be satisfied:

$$F_{0ij}(t) = \left\|(\mathbf{x}_{0i} + t\mathbf{v}_{0i}) \times (\mathbf{x}_{ji} + t\mathbf{v}_{ji})\right\|_1 - S\left\|\mathbf{x}_{ji} + t\mathbf{v}_{ji}\right\|_\infty \leq 0 \quad (1)$$

in which $S = 3R$. So instead of finding t_0^{E} that minimizes the Euclidean distance, we define the time t_0 as the local minimum of $F_{0ij}(t)$ closest to t_0^{E}, which may or may not be the global minimum. Since $F_{0ij}(t_0^{\mathrm{E}}) \leq 0$ when a collision occurs, we must have $F_{0ij}(t_0) \leq 0$. In addition, we need to consider the boundary case when the vertex-edge distance is not the vertex-line distance (i.e., when the vertex's projection on the edge line is outside of the edge). By formulating the test into a constrained optimization problem, we must be able to find t_0 from five Karush–Kuhn–Tucker cases.

Case 1. $t_0 = 0$ or 1.

Algorithm 2 VE_CCD($\mathbf{x}_0^0, \mathbf{x}_0^1, \mathbf{x}_i^0, \mathbf{x}_i^1, \mathbf{x}_j^0, \mathbf{x}_j^1, R$)

```
Compute $\{\mathbf{x}_{0i}, \mathbf{x}_{ij}\}$ and $\{\mathbf{v}_{0i}, \mathbf{v}_{ij}\}$;
$\mathbf{T} \leftarrow \left\{0, 1, -\frac{x_{ji} \pm y_{ji}}{u_{ji} \pm v_{ji}}, -\frac{x_{ji} \pm z_{ji}}{u_{ji} \pm w_{ji}}, -\frac{z_{ji} \pm y_{ji}}{w_{ji} \pm v_{ji}}\right\}$;        ▷ Case 1 and 3
$\mathbf{x}_a \leftarrow \mathbf{v}_{0i} \times \mathbf{v}_{ji}$;
$\mathbf{x}_b \leftarrow \mathbf{x}_{0i} \times \mathbf{v}_{ji} + \mathbf{v}_{0i} \times \mathbf{x}_{ji}$;
$\mathbf{x}_c \leftarrow \mathbf{x}_{0i} \times \mathbf{x}_{ji}$;
for every component $k$ in $\mathbf{x}_a$ and $\mathbf{x}_b$ do
    if $|k| \le \rho$ then $k \leftarrow 0$;
$\mathbf{T} \leftarrow \mathbf{T} \bigcup$ Quadratic_Solver($\mathbf{x}_a.x, \mathbf{x}_b.x, \mathbf{x}_c.x$);        ▷ Case 4
$\mathbf{T} \leftarrow \mathbf{T} \bigcup$ Quadratic_Solver($\mathbf{x}_a.y, \mathbf{x}_b.y, \mathbf{x}_c.y$);
$\mathbf{T} \leftarrow \mathbf{T} \bigcup$ Quadratic_Solver($\mathbf{x}_a.z, \mathbf{x}_b.z, \mathbf{x}_c.z$);
for every $at^2 + bt + c = 0$ in Case 2 do
    $\mathbf{T} \leftarrow \mathbf{T} \bigcup \{-b/2a\}$;        ▷ Case 2
for every root $t \in \mathbf{T}$ do        ▷ Proximity test
    Compute $\mathbf{x}_{0i}^t, \mathbf{x}_{ji}^t$;        ▷ $\mathbf{x}_{0i}^t \leftarrow \mathbf{x}_{0i} + t\mathbf{v}_{0i}$
    if $0 \le t \le 1$ and $\mathbf{x}_{ji}^t \cdot \mathbf{x}_{ji}^t \ge \lambda^2$ and $0 \le \mathbf{x}_{0i}^t \cdot \mathbf{x}_{ji}^t \le \mathbf{x}_{ji}^t \cdot \mathbf{x}_{ji}^t$ and
      $F_{0ij}(t) \le \psi$ then return true;
if VV_CCD($\mathbf{x}_0^0, \mathbf{x}_0^1, \mathbf{x}_i^0, \mathbf{x}_i^1, D^2$) or
    VV_CCD($\mathbf{x}_0^0, \mathbf{x}_0^1, \mathbf{x}_j^0, \mathbf{x}_j^1, D^2$) then return true;
return false;
```

Case 2. The components in $\mathbf{x}_{ji}+t_0\mathbf{v}_{ji}$ have unique magnitudes and no component in $(\mathbf{x}_{0i} + t_0\mathbf{v}_{0i}) \times (\mathbf{x}_{ji} + t_0\mathbf{v}_{ji})$ is equal to zero. In this case, $F_{0ij}(t_0)$ can be considered as a quadratic function locally at t_0 and the solution of $F'_{0ij} = 0$ gives t_0. There are eight possibilities in $\left\|(\mathbf{x}_{0i} + t\mathbf{v}_{0i}) \times (\mathbf{x}_{ji} + t\mathbf{v}_{ji})\right\|_1$ and six possibilities in $\left\|\mathbf{x}_{ji} + t\mathbf{v}_{ji}\right\|_\infty$. Due to the symmetry, only 24 time candidates are unique.

Case 3. At least two components of $\mathbf{x}_{ji} + t_0\mathbf{v}_{ji}$ have the same magnitude at time t_0. If so, we find t_0 by solving a linear equation. Since there are three components and each component can be either positive or negative, there are at most six unique candidates.

Case 4. At least one component of $(\mathbf{x}_{0i}+t_0\mathbf{v}_{0i})\times(\mathbf{x}_{ji}+t_0\mathbf{v}_{ji})$ is zero at time t_0. If so, we can find t_0 by solving the quadratic equation. There are three equations and six collision time candidates at most.

Case 5. The time t_0 is the vertex-vertex collision time between $\mathbf{x}_0$ and $\mathbf{x}_i$, or $\mathbf{x}_0$ and $\mathbf{x}_j$.

Degenerate cases. Although degenerate cases are rare, we still need to examine them to ensure the robustness of vertex-edge C-CD. First of all, any linear equation may be trivially reduced to a constant in Case 3. If so, t_0 has no influence on $\left\|\mathbf{x}_{ji} + t\mathbf{v}_{ji}\right\|_\infty$ and we can find it from other cases. Similarly, if any quadratic equation[1] in Case 4 is reduced to a constant, we find t_0 from other cases. If all equations in Case 3 and 4 are trivial, the choice of t has no influence on $F_{0ij}(t)$ and we can simply set $t_0 = 0$ or 1. This may happen when the vertex-line distance is relatively static. If the quadratic equation in Case 2 is reduced to a linear equation or even a constant, then $F_{0ij}(t)$ must be minimized at $t_0 = 0$ or 1, which is covered by Case 1. We note that we have not discussed $\left\|\mathbf{x}_{ji} + t_0\mathbf{v}_{ji}\right\|_\infty = 0$, which makes Equation 1 ineffective. We will study it in Subsection 4.3.

Summary. Given the five cases, we can formulate our vertex-edge CCD algorithm in two steps as Algorithm 2 shows. In the first step, it computes a set of collision time candidates. In the second step, it verifies whether $F_{0ij}(t)$ is indeed small at each candidate. Although there are many candidates, only a few are valid (between 0 and 1) and need further verification. The average number of valid candidates is about seven as our experiment shows.

[1]Our quadratic solver (in Appendix A) can handle the degenerate case when a quadratic equation is reduced to a linear equation.

One question is how to determine the distance threshold D to capture missed collision events by vertex-vertex CCD. Let t_0 be the time when $F_{0ij}(t)$ is minimized and the time t_0^{E} be the time when the Euclidean distance is minimized. If the vertex's projection is outside of the edge at t_0^{E}, we can set $D \ge R$. If the vertex's projection is inside of the edge at t_0, vertex-vertex CCD is unused. So collisions are missed when the vertex's projection is outside at t_0 but inside at t_0^{E} only, as Figure 3a shows. Fortunately, since t_0 is the minimum of $F_{0ij}(t)$ closest to t_0^{E}, $F_{0ij}(t) \le 0$ and the vertex-line distance must be shorter than S at any $t \in [t_0, t_0^{\mathrm{E}}]$. When the vertex's projection moves from the interior of the edge to the exterior, there must exist time t when the vertex's projection is at an endpoint. We use $D \ge S = 3R$ to capture this collision by vertex-vertex CCD at t.

3.3 Vertex-Triangle Collision Detection

In our vertex-triangle CCD algorithm shown in Algorithm 3, we apply vertex-edge CCD after the cubic-based test [Provot 1997; Bridson et al. 2002], to avoid missed collisions. The closed-form solution of a cubic equation requires the cubic root, which cannot be correctly rounded under the IEEE 754 standard. Instead, a numerical solver finds $\hat{t}_0$, such that the residual error at $\hat{t}_0$ is below a threshold μ. The use of this threshold[2] suffers from two problems.

Small triangle. The cubic function $F(t) = (\mathbf{x}_{01} + t\mathbf{v}_{01}) \cdot (\mathbf{x}_{21} + t\mathbf{v}_{21}) \times (\mathbf{x}_{31} + t\mathbf{v}_{31})$ is proportional to the Euclidean vertex-plane distance times the triangle area at time t. So if the triangle is small at $\hat{t}_0$, the residual error is also small, even when the vertex-plane distance is still large. As a result, proximity test will fail at time $\hat{t}_0$.

Parallel motion. Another problem happens when the vertex moves nearly parallel to the triangle plane, in which case the residual error becomes small even before the vertex intersects the triangle.

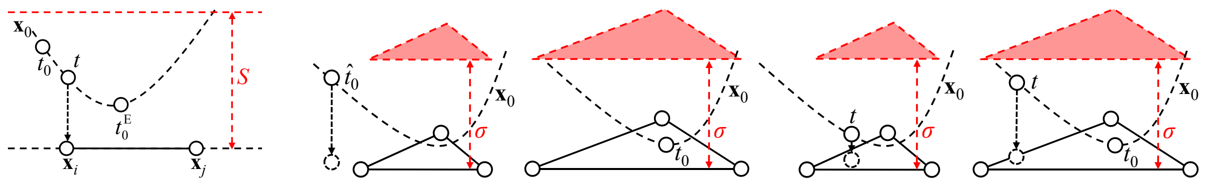
Due to the motion of the primitives, it is possible to notice such degenerate problems only at $\hat{t}_0$ and in its neighborhood. So we cannot detect them as special cases without computing $\hat{t}_0$. To prevent them from causing collision failures, we apply vertex-edge CCD after proximity test. If the triangle is small when a collision happens, the vertex must be close to a triangle edge and this collision can be detected by vertex-edge CCD. Meanwhile, if a vertex moves nearly parallel to the triangle plane, it must get sufficiently close to a triangle edge before it intersects the triangle. So vertex-edge CCD can detect this collision as well.

3.3.1 Errors and False Positives

Here we would like to study the overall error without considering the rounding errors. Our goal is: 1) to eliminate false negatives; and 2) to reduce false positives as much as possible.

Let μ be the convergence threshold of the cubic solver, σ be the thickness threshold in the vertex-triangle proximity test, and R be the Euclidean distance threshold in vertex-edge CCD. Let t_0 be the exact vertex-triangle intersection time (such that $F(t_0) = 0$) and $\hat{t}_0$ be its computed version (such that $|F(\hat{t}_0)| \le \mu$). By definition, $F(t) = A(t)C(t)$, in which $A(t)$ is twice the triangle area and $C(t)$ is the vertex-plane distance. Since $|F(\hat{t}_0)| \le \mu$, instead of testing whether $|C(\hat{t}_0)| \le \sigma$, we test whether $|A(\hat{t}_0)| \ge \tau = \mu/\sigma$. If so, we have $|C(\hat{t}_0)| \le \sigma$ and the test proceeds. If not, we do vertex-edge CCD. Assuming that $|F(t)| \le \mu$ for any t in $[t_0, \hat{t}_0]$ (in Appendix B), we claim every exact collision event will be detected by setting $R = \mu^{\frac{1}{3}}$. To prove this, we show that if an exact collision fails the vertex-triangle proximity test, it must pass vertex-edge CCD.

[2]A numerical solver may use other convergence criteria, but they are ineffective once the rounding error becomes dominant.



(a) Vertex-edge case (b) $A(\hat{t}_0) \le \tau$ (c) $A(t_0) \ge \tau$ (d) $A(t) = \tau$ (e) Projected onto an edge

Figure 3: *Vertex-edge and vertex-triangle CCD cases. When the vertex* $\mathbf{x}_0$ *moves along its trajectory (in a dashed line) over time, the edge length and the triangle size may vary. The red lines and triangles represent the upper bounds on the vertex-line and vertex-plane distances.*

Algorithm 3 VT_CCD($\mathbf{x}_0^0, \mathbf{x}_0^1, \mathbf{x}_1^0, \mathbf{x}_1^1, \mathbf{x}_2^0, \mathbf{x}_2^1, \mathbf{x}_3^0, \mathbf{x}_3^1$)

```
Compute $\{\mathbf{x}_{01}, \mathbf{x}_{21}, \mathbf{x}_{31}\}$ and $\{\mathbf{v}_{01}, \mathbf{v}_{21}, \mathbf{v}_{31}\}$;
$a \leftarrow \mathbf{v}_{01} \cdot \mathbf{v}_{21} \times \mathbf{v}_{31}$;
$b \leftarrow \mathbf{x}_{01} \cdot \mathbf{v}_{21} \times \mathbf{v}_{31} + \mathbf{v}_{01} \cdot \mathbf{v}_{21} \times \mathbf{x}_{31} + \mathbf{v}_{01} \cdot \mathbf{x}_{21} \times \mathbf{v}_{31}$;
$c \leftarrow \mathbf{x}_{01} \cdot \mathbf{v}_{21} \times \mathbf{x}_{31} + \mathbf{x}_{01} \cdot \mathbf{x}_{21} \times \mathbf{v}_{31} + \mathbf{v}_{01} \cdot \mathbf{x}_{21} \times \mathbf{x}_{31}$;
$d \leftarrow \mathbf{x}_{01} \cdot \mathbf{x}_{21} \times \mathbf{x}_{31}$;
$\mathbf{T} \leftarrow \{0, 1\} \bigcup$ Cubic_Solver($a, b, c, d, \mu$);
for every root $t \in \mathbf{T}$ do                                ▹ Proximity test
    Compute $\mathbf{x}_{01}^t, \mathbf{x}_{21}^t, \mathbf{x}_{31}^t$;                 ▹ $\mathbf{x}_{01}^t \leftarrow \mathbf{x}_{01} + t\mathbf{v}_{01}$
    $\mathbf{N} = \mathbf{x}_{21}^t \times \mathbf{x}_{31}^t$;
    if $\|\mathbf{N}\|_\infty \ge \tau$ and
        $\max\left(\left\|\mathbf{x}_{21}^t\right\|_\infty, \left\|\mathbf{x}_{31}^t\right\|_\infty, \left\|\mathbf{x}_{31}^t - \mathbf{x}_{21}^t\right\|_\infty\right)\eta \le \|\mathbf{N}\|_\infty$ and
        $0 \le \mathbf{N} \cdot \mathbf{x}_{21}^t \times \mathbf{x}_{01}^t \le \mathbf{N} \cdot \mathbf{N}$ and
        $0 \le \mathbf{N} \cdot \mathbf{x}_{01}^t \times \mathbf{x}_{31}^t \le \mathbf{N} \cdot \mathbf{N}$ and
        $0 \le \mathbf{N} \cdot \mathbf{x}_{21}^t \times \mathbf{x}_{01}^t + \mathbf{N} \cdot \mathbf{x}_{01}^t \times \mathbf{x}_{31}^t \le \mathbf{N} \cdot \mathbf{N}$ then return true;
if VE_CCD($\mathbf{x}_0^0, \mathbf{x}_0^1, \mathbf{x}_1^0, \mathbf{x}_1^1, \mathbf{x}_2^0, \mathbf{x}_2^1, R$) or       ▹ /* Additional tests */
  VE_CCD($\mathbf{x}_0^0, \mathbf{x}_0^1, \mathbf{x}_2^0, \mathbf{x}_2^1, \mathbf{x}_3^0, \mathbf{x}_3^1, R$) or
  VE_CCD($\mathbf{x}_0^0, \mathbf{x}_0^1, \mathbf{x}_3^0, \mathbf{x}_3^1, \mathbf{x}_1^0, \mathbf{x}_1^1, R$) then return true;
return false;
```

Algorithm 4 EE_CCD($\mathbf{x}_0^0, \mathbf{x}_0^1, \mathbf{x}_1^0, \mathbf{x}_1^1, \mathbf{x}_2^0, \mathbf{x}_2^1, \mathbf{x}_3^0, \mathbf{x}_3^1$)

```
Compute $\{\mathbf{x}_{20}, \mathbf{x}_{10}, \mathbf{x}_{32}\}, \{\mathbf{v}_{20}, \mathbf{v}_{10}, \mathbf{v}_{32}\}$;
$a \leftarrow \mathbf{v}_{20} \cdot \mathbf{v}_{10} \times \mathbf{v}_{32}$
$b \leftarrow \mathbf{x}_{20} \cdot \mathbf{v}_{10} \times \mathbf{v}_{32} + \mathbf{v}_{20} \cdot \mathbf{v}_{10} \times \mathbf{x}_{32} + \mathbf{v}_{20} \cdot \mathbf{x}_{10} \times \mathbf{v}_{32}$;
$c \leftarrow \mathbf{x}_{20} \cdot \mathbf{v}_{10} \times \mathbf{x}_{32} + \mathbf{x}_{20} \cdot \mathbf{x}_{10} \times \mathbf{v}_{32} + \mathbf{v}_{20} \cdot \mathbf{x}_{10} \times \mathbf{x}_{32}$;
$d \leftarrow \mathbf{x}_{20} \cdot \mathbf{x}_{10} \times \mathbf{x}_{32}$;
$\mathbf{T} \leftarrow \{0, 1\} \bigcup$ Cubic_Solver($a, b, c, d, \mu$);
for every root $t \in \mathbf{T}$ do                                ▹ Proximity test
    Compute $\mathbf{x}_{20}^t, \mathbf{x}_{10}^t, \mathbf{x}_{32}^t$;                 ▹ $\mathbf{x}_{20}^t \leftarrow \mathbf{x}_{20} + t\mathbf{v}_{20}$
    $\mathbf{N} = \mathbf{x}_{10}^t \times \mathbf{x}_{32}^t$;
    if $\|\mathbf{N}\|_\infty \ge \tau$ and $\max(\left\|\mathbf{x}_{10}^t\right\|_\infty, \left\|\mathbf{x}_{32}^t\right\|_\infty) \cdot \eta \le \|\mathbf{N}\|_\infty$ and
      $0 \le \mathbf{N} \cdot \mathbf{x}_{20}^t \times \mathbf{x}_{10}^t \le \mathbf{N} \cdot \mathbf{N}$ and
      $0 \le \mathbf{N} \cdot \mathbf{x}_{20}^t \times \mathbf{x}_{32}^t \le \mathbf{N} \cdot \mathbf{N}$ then return true;
if VE_CCD($\mathbf{x}_2^0, \mathbf{x}_2^1, \mathbf{x}_0^0, \mathbf{x}_0^1, \mathbf{x}_1^0, \mathbf{x}_1^1, R$) or         ▹ Additional tests
  VE_CCD($\mathbf{x}_3^0, \mathbf{x}_3^1, \mathbf{x}_0^0, \mathbf{x}_0^1, \mathbf{x}_1^0, \mathbf{x}_1^1, R$) or
  VE_CCD($\mathbf{x}_0^0, \mathbf{x}_0^1, \mathbf{x}_2^0, \mathbf{x}_2^1, \mathbf{x}_3^0, \mathbf{x}_3^1, R$) or
  VE_CCD($\mathbf{x}_1^0, \mathbf{x}_1^1, \mathbf{x}_2^0, \mathbf{x}_2^1, \mathbf{x}_3^0, \mathbf{x}_3^1, R$) then return true;
return false;
```

Triangle area test. The proximity test fails because $A(\hat{t}_0) \le \tau$ at time $\hat{t}_0$, as Figure 3b shows. If we also have $A(t_0) \le \tau$, then the maximum squared radius of the triangle inner circle is $\frac{\sqrt{3}}{18}\tau$ and at least one squared Euclidian vertex-edge distance is less than $\frac{\sqrt{3}}{18}\tau$ at time t_0. So by setting $R^2 \ge \frac{\sqrt{3}}{18}\tau$, we can get this collision reported by vertex-edge CCD. On the other hand, we may have $A(t_0) \ge \tau$ (Figure 3c). Since $A(t)$ and the vertex's projection are two continuous functions of time, there must exist time t in $[t_0, \hat{t}_0]$, such that $A(t) = \tau$ and the vertex's projection is within the triangle (Figure 3d), or $A(t) \ge \tau$ and the vertex's projection is on a triangle edge (Figure 3e), the latter of which may not happen if the vertex's projection is always within the triangle from t_0 to $\hat{t}_0$. If the first event happens, we have $C(t) \le \mu/\tau = \sigma$ and we can set $R^2 \ge \sigma^2 + \frac{\sqrt{3}}{18}\tau$ to detect this case at time t. If the second event happens, the vertex-edge distance is less than σ at time t and we need $R \ge \sigma$. We will refer to the idea of defining two events and finding the time when an event happens as the *two-event analysis* in the rest of this paper.

One interesting observation from the above analysis is that if $A(t) \le \tau$ at any $t \in [t_0, \hat{t}_0]$, we can replace $\hat{t}_0$ by t and use the above analysis to conclude that $R^2 \ge \sigma^2 + \frac{\sqrt{3}}{18}\tau$ is also sufficient to get such collisions detected. So we need to consider only the cases when $A(t) \ge \tau$ for any $t \in [t_0, \hat{t}_0]$ next.

Barycentric test. If $A(\hat{t}_0) > \tau$, the proximity test can fail only because of the barycentric test, in which case the vertex's projection is outside of the triangle at time $\hat{t}_0$. Let t be the time when the vertex's projection intersects a triangle edge. Since $A(t) \ge \tau$ according to the previous analysis, the vertex-plane distance is bounded by σ and we can use $R \ge \sigma$ to detect such collisions at time t.

Summary. In conclusion, $R^2 \ge \sigma^2 + \frac{\sqrt{3}}{18}\tau$ is sufficient to detect missed collision events by vertex-edge CCD. A simple way to reduce the false positives is make R as small as possible. Given $\tau = \mu/\sigma$, we can minimize R by setting $\tau^3 = \frac{36\mu^2}{\sqrt{3}}$, in which case $R^3 = \frac{1}{4}\mu$. For simplicity, we use $\tau = 4\mu^{\frac{2}{3}}$ and $R = \mu^{\frac{1}{3}}$. We will discuss the changes caused by the rounding errors in Section 4.4.

3.4 Edge-Edge Collision Detection

To detect the collision between two edges $\mathbf{x}_0\mathbf{x}_1$ and $\mathbf{x}_2\mathbf{x}_3$, we formulate edge-edge CCD similar to vertex-triangle CCD, except that the normal $\mathbf{N}$ is defined as the cross product of the two edges. In this case, $\|\mathbf{N}\|_2$ is equal to twice the area of the quadrilateral formed by the four vertex projections along the $\mathbf{N}$ direction, and the cubic function is defined as: $F(t) = (\mathbf{x}_{20} + t\mathbf{v}_{20}) \cdot (\mathbf{x}_{10} + t\mathbf{v}_{10}) \times (\mathbf{x}_{32} + t\mathbf{v}_{32})$. The algorithm suffers from similar problems: *small quadrilateral*, when the quadrilateral formed by the two edges is small; and *perpendicular motion*, when the relative motion of the edges is nearly perpendicular to $\mathbf{N}$. Our solution again is to use vertex-edge CCD.

3.4.1 Errors and False Positives

By ignoring the rounding errors, we would like to: 1) to eliminate false negatives; and 2) to reduce false positives. Let μ be the convergence threshold of the cubic solver, σ be the thickness threshold on the line-line distance, and R be the distance threshold in vertex-edge CCD. By definition, $F(t) = A(t)C(t)$, in which $A(t)$ is twice the quadrilateral area and $C(t)$ is the line-line distance. Similar to vertex-triangle CCD, the proximity test contains two steps: the area test: $|A(\hat{t}_0)| \ge \tau = \mu/\sigma$; and the barycentric test. If a real collision fails any step, we must use vertex-edge CCD to detect it.

Quadrilateral area test. The proximity test fails because $A(\hat{t}_0) \leq \tau$ at time $\hat{t}_0$. Given a quadrilateral defined by two crossing edges, the shortest distance from any vertex to the other edge must be bounded by half of the shorter edge length. If $A(t_0) \leq \tau$, the shorter crossing edge length cannot exceed $\sqrt{\tau}$, and the shortest Euclidean vertex-edge distance is less than $\frac{\sqrt{\tau}}{2}$ at time t_0. This means we can use vertex-edge CCD to detect such events by setting $R^2 \geq \frac{\tau}{4}$. On the other hand, we may have $A(t_0) \geq \tau$. Using the two-event analysis presented in Subsection 3.3.1, we formulate a sufficient condition to get the collision detected: $R^2 \geq \sigma^2 + \frac{\tau}{4}$. Based on the same analysis in Subsection 3.3.1, we know $R^2 \geq \sigma^2 + \frac{\tau}{4}$ is also sufficient to detect any collision, if $A(t) \leq \tau$ happens at any $t \in [t_0, \hat{t}_0]$.

Barycentric test. When the barycentric test fails, the edges do not intersect at time $\hat{t}_0$ in the projected plane. Let t be the time when a projected edge intersects the other projected edge's vertex. As shown previously, we have $A(t) \geq \tau$ and the line-line distance is bounded by σ at t. So $R \geq \sigma$ is sufficient.

Summary. In conclusion, $R^2 \geq \sigma^2 + \frac{\tau}{4}$ is sufficient to prevent collision events from being missed. To reduce the false positives, we need to make R as small as possible. Given $\tau = \mu/\sigma$, we can minimize R by setting $\tau^3 = 8\mu^2$, in which case $R^2 = \frac{3}{4}\mu^{\frac{2}{3}}$. For simplicity, we use $\tau = 2\mu^{\frac{2}{3}}$ and $R = \mu^{\frac{1}{3}}$.

4 Error Analysis

Compared with the numerical errors, the rounding errors caused by floating-point arithmetic are more complex. We will study their properties in Subsection 4.1 and then discuss their effect in CCD.

4.1 Rounding Error Properties

Let a be a real number, the rounding process converts a into a floating-point number $\hat{a}$, such that $|a - \hat{a}| < |a|\epsilon$, in which ϵ is *machine epsilon*. The single precision floating-point format uses $\epsilon = 2^{-23}$, and the double precision format uses $\epsilon = 2^{-52}$. The arithmetic operations performed on floating-point numbers are *floating-point arithmetic*. According to the IEEE 754 standard, basic floating-point arithmetic operations, including `add`, `subtract`, `multiply`, `divide`, and `square_root`, must be correctly rounded. In other words, given an exact arithmetic operator $*$ and its floating-point counterpart $\circledast$, $a \circledast b = \mathtt{ROUND}(a*b)$ and $|a \circledast b - a*b| \leq |a*b|\epsilon$.

When there are multiple floating-point operations, rounding errors are accumulated and the error bounds are not straightforward to compute. To handle this issue, we propose an error estimation strategy based on forward error analysis. Let f be a function containing multiple arithmetic operations. Its rounding error $|\hat{f} - f|$ is bounded by: $E_f = B_f((1+\epsilon)^{k_f} - 1)$, in which B_f is an upper bound on $|f|$ and k_f is an exponential coefficient. Assuming that all variables in f are bounded by B_0, we can formulate the operations of f into a tree and find B_f and k_f as:

- If node i is a leaf, $B_i = B_0$ and $k_i = 0$.
- If i is a non-leaf node corresponding to `add` or `subtract` and it has two children l and r, then $B_i = B_l + B_r$ and $k_i = \max(k_l, k_r) + 1$.
- If i is a non-leaf node corresponding to `multiply` and it has two children l and r, then $B_i = B_l B_r$ and $k_i = k_l + k_r + 1$.

We prove the correctness of these rules by the following theorem.

Theorem 4.1. *If f is a function made of* `add`*,* `subtract`*, and* `multiply` *operations, then $|f| \leq B_f$ and $|f - \hat{f}| \leq E_f = B_f((1+\epsilon)^{k_f} - 1)$, in which B_f and k_f are calculated using the above rules.*

Proof. This can be proven by induction. By definition, a leaf node is an input variable, so its bound is B_0 and its error bound is 0. Now suppose that B_l, k_l, B_r, and k_r can provide valid bounds for the children of a non-leaf node. If the node i corresponds to `add` or `subtract`, then $|i| = |l \pm r| \leq |l| + |r| \leq B_l + B_r$, and $|\hat{i} - i| \leq \left|\hat{l} - l\right| + |\hat{r} - r| + \left|\hat{i} - (\hat{l} \pm \hat{r})\right| \leq B_i((1+\epsilon)^{k_i - 1} - 1) + B_i(1+\epsilon)^{k_i-1}\epsilon = B_i((1+\epsilon)^{k_i} - 1)$. If i corresponds to `multiply`, then $|i| = |lr| \leq B_l B_r$ and $|\hat{i} - i| \leq \left|\hat{l}\hat{r} - lr\right| + \left|\hat{i} - \hat{l}\hat{r}\right| \leq B_i((1+\epsilon)^{k_l + k_r} - 1) + B_i(1+\epsilon)^{k_l + k_r}\epsilon = B_i((1+\epsilon)^{k_i} - 1)$. So B_i and k_i are valid for node i as well. □

The Upper Bound B. Our algorithms start with vector computation to obtain relative positions and velocities. To apply Theorem 4.1 in error analysis, we would like to derive an upper bound on their vector components. Here we consider the cases when exact collisions happen only. Let $E_{\max}$ be an upper bound on the edge vectors at time 0: $E^0_{\max} = \max_{\{i,j\}} \left\|\mathbf{x}^0_i - \mathbf{x}^0_j\right\|_\infty$ for any edge $\{i, j\}$. Let $V_{\max}$ be an upper bound on the velocities: $V_{\max} = \max_i \left\|\mathbf{x}^1_i - \mathbf{x}^0_i\right\|_\infty$. If the vertex $\mathbf{x}_0$ collides with the triangle at a point $\mathbf{x}^t_m$ when $t \in [0, 1]$, then their distance at time 0 must be bounded by $2V_{\max}$. Since the distance from $\mathbf{x}^0_m$ to any triangle vertex is bounded by $E^0_{\max}$, we know $\mathbf{x}^0_{01}$, $\mathbf{x}^0_{02}$, and $\mathbf{x}^0_{03}$ is bounded by $E^0_{\max} + 2V_{\max}$. By definition, all of the relative velocities are bounded by $2V_{\max}$. So $B = E^0_{\max} + 2V_{\max}$ is an upper bound on the vector components of both the displacements and velocities in vertex-triangle CCD. The same analysis can be applied to edge-edge CCD, except that collision can happen anywhere within the two edges. Because of that, there must exist vertex i from one edge and vertex j from the other edge, such that $B = E^0_{\max} + 2V_{\max}$ is an upper bound on the displacement from i to j. In practice, this vertex pair can be detected as the one with the minimum component, and we assume $\mathbf{x}_{20}$ is such a pair in Algorithm 4. We note that B is also an upper bound on the relative positions at any time in $[0, 1]$. We will use this fact to study the errors of proximity tests later. A small issue is B cannot be exactly computed. According to Theorem 4.1, the error involved in B is $B((1+\epsilon)^2 - 1)$ at most, due to two `subtract` and one `add`. So to make B valid in both exact and computed cases, we set: $B = (E^0_{\max} + 2V_{\max})(1+\epsilon)^2$. When the primitives move fast, we make B tighter by using relative velocities. Specifically, we compute the bounding boxes at the starting and ending times, and use their difference to obtain an upper bound $V^{\text{rel}}_{\max}$ on the relative velocities, similar to [Selle et al. 2009]. We then set $V_{\max}$ to $V^{\text{rel}}_{\max}$, if $V^{\text{rel}}_{\max}$ is smaller.

For simplicity, we ignore the errors of the relative positions and velocities in this section. We will visit them later in Section 5.

4.2 Errors in Vertex-Vertex CCD

We first study the collision time t_0 in vertex-vertex CCD. Let $\left\|\mathbf{x}_{ji}\right\|_\infty \leq B$, $\left\|\mathbf{v}_{ji}\right\|_\infty = b \leq B$, $m = -\mathbf{x}_{ji} \cdot \mathbf{v}_{ji}$ and $n = \mathbf{v}_{ji} \cdot \mathbf{v}_{ji}$. We know from Theorem 4.1 and Figure 4a that $|\hat{m} - m| \leq 3Bb((1+\epsilon)^3 - 1)$ and $|\hat{n} - n| \leq 3b^2((1+\epsilon)^3 - 1)$. Given $n \geq b^2$, we have: $\left|\frac{\hat{m}}{\hat{n}} - \frac{m}{n}\right| \leq \left|\frac{\hat{m}-m}{\hat{n}}\right| + t_0\left|\frac{\hat{n}-n}{\hat{n}}\right| \leq 10\epsilon\left(\frac{B}{b} + 1\right)$. Since $t_0 \in [0, 1]$, $|\hat{t}_0 - t_0| \leq \left|\hat{t}_0 - \frac{\hat{m}}{\hat{n}}\right| + \left|\frac{\hat{m}}{\hat{n}} - \frac{m}{n}\right| \leq \left(1 + 10\epsilon\left(\frac{B}{b} + 1\right)\right)\epsilon + 10\epsilon\left(\frac{B}{b} + 1\right) \leq 12\epsilon\left(\frac{B}{b} + 1\right)$. This conclusion is still valid after applying `Clamp` on $\hat{t}$, which makes $|\hat{t}_0 - t_0|$ even smaller. Let $F_{ij}(t) = \left\|\mathbf{x}_{ij} + t\mathbf{v}_{ij}\right\|_2$ be the Euclidean distance at time t. Since $\left\|\mathbf{v}_{ji}\right\|_\infty = b \leq B$, we must have $\left\|\mathbf{v}_{ji}\right\|_2 \leq \sqrt{3}b$ and $\left|F_{ij}(\hat{t}) - F_{ij}(t)\right| \leq \sqrt{3}b12\epsilon\left(\frac{B}{b} + 1\right) \leq 24\sqrt{3}B\epsilon$. Meanwhile, $\left|F_{ij}(t)\right| \leq \left\|\mathbf{x}_{ij}\right\|_2 + \left\|\mathbf{v}_{ij}\right\|_2 \leq 2\sqrt{3}B$, so $\left|F^2_{ij}(\hat{t}) - F^2_{ij}(t)\right| \leq (4\sqrt{3}B + 24\sqrt{3}B\epsilon) \cdot 24\sqrt{3}B\epsilon \leq 289B^2\epsilon$.

The proximity test cannot evaluate $F^2_{ij}(\hat{t})$ exactly. Since the components of $\mathbf{x}_{ji}$, $\mathbf{v}_{ji}$, and $\mathbf{x}_{ji} + \hat{t}\mathbf{v}_{ji}$ are all bounded by B, the error associated with the components of $\mathbf{x}_{ji} + \hat{t}\mathbf{v}_{ji}$ must be bounded by $(B + B\epsilon)\epsilon + B\epsilon \leq B((1+\epsilon)^2 - 1)$. Using Theorem 4.1, we can bound the proximity test error by: $\left|\hat{F}^2_{ij}(\hat{t}) - F^2_{ij}(\hat{t})\right| \leq 3B^2((1+\epsilon)^7 - 1) \leq$

$22B^2\epsilon$. Together, we have: $\left|\hat{F}_{ij}^2(\hat{t}) - F_{ij}^2(t)\right| \leq 311B^2\epsilon$ and we can set the error tolerance $\delta = 311B^2\epsilon$ to avoid false negatives.

False positives. When a false positive occurs, We must have: $\left|F_{ij}^2(t) - D^2\right| \leq 2\delta$. Assuming that D^2 is substantially larger than 2δ, we get the distance error: $\left|F_{ij}(t) - D\right| \leq \sqrt{2\delta} \leq 25B\epsilon^{\frac{1}{2}}$.

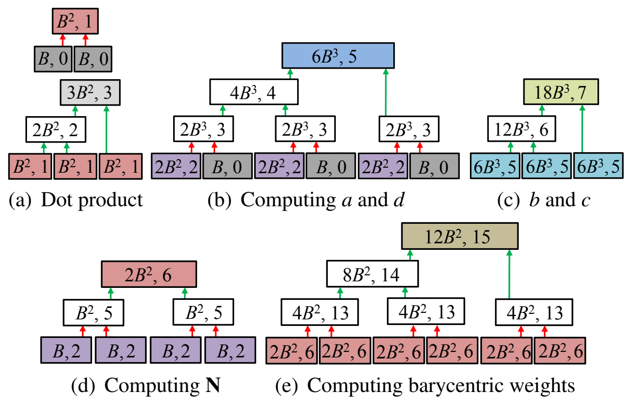
4.3 Errors in Vertex-Edge CCD

We first study the error of $F_{0ij}(t)$, which is related to the vertex-line distance. To simplify the analysis, we modify $F_{0ij}(t)$ by removing any component in $\mathbf{x}_a$ or $\mathbf{x}_b$, if it is less than $\rho = 3B^2\epsilon^{\frac{1}{2}}$. Let the result be $G_{0ij}(t)$. Since $|F_{0ij}(t) - G_{0ij}(t)|$ is small, the minimization of $G_{0ij}(t)$ approximates the minimization of $F_{0ij}(t)$ well.

Case 1 to 4. According to Algorithm 2, there are four cases related to $G_{0ij}(t)$. Let t_0^{F} be the local minimum of $F_{0ij}(t)$ closest to t_0^{E}, t_0 be the local minimum of $G_{0ij}(t)$ closest to t_0^{F}, and $\hat{t}_0$ be the computed version of t_0. In the first three cases, $\hat{t}_0$ is given by simple linear equations and it is not difficult to see that $|G_{0ij}(\hat{t}_0) - G_{0ij}(t_0)|$ and $F_{0ij}(\hat{t}_0) - F_{0ij}(t_0^{\mathrm{F}})$ are bounded by $B^2O(\epsilon)$. The problem comes from Case 4, which requires to solve quadratic equations. Let $Q(t) = at^2 + bt + c = 0$ be the exact quadratic equation, and $\bar{Q}(\bar{t}) = \hat{a}t^2 + \hat{b}t + \hat{c}$ be the actual equation sent to the solver. The time $\hat{t}_0$ is computed from $\bar{Q}(\bar{t})$, not $Q(t)$. Since $|\hat{a} - a|$, $|\hat{b} - b|$, and $|\hat{c} - c|$ are small, the residual error $|Q(\hat{t}_0) - Q(t_0)|$ is also small. Unfortunately, $|\hat{t}_0 - t_0|$ can be large and it can cause $|G_{0ij}(\hat{t}_0) - G_{0ij}(t_0)|$ to be as large as $688B^2\epsilon^{\frac{1}{2}}$. In our detailed analysis in Appendix C, we further show $F_{0ij}(t) - F_{0ij}(t_0^{\mathrm{F}}) \leq 725B^2\epsilon^{\frac{1}{2}}$ for any $t \in [t_0^{\mathrm{F}}, \hat{t}_0]$. So we set the tolerance threshold $\psi = 725B^2\epsilon^{\frac{1}{2}}$ in Algorithm 2, to avoid failures due to the rounding errors.

Case 5. Case 5 is done by vertex-vertex CCD and our job is to find a suitable distance threshold D in vertex-vertex CCD. Let t_0^{E} be the exact time when the Euclidean vertex-edge distance gets minimized. We need to study only the cases when the vertex's projection is inside of the edge at t_0^{E}, as discussed in Subsection 3.2. To further reduce false positives, we introduce an edge length test. Our goal is to make sure if a real collision fails both the length test and the barycentric test, it must be detected by vertex-vertex CCD.

- **Edge length test.** This test prevents $F_{0ij}(t)$ from inviting large false positives when the edge is short: $\hat{l}^2 \leq \lambda^2$, in which $\hat{l}$ is the computed Euclidean edge length at time $\hat{t}_0$. Since $|\hat{l}^2 - l^2|$ is at most $3B^2((1+\epsilon)^7 - 1) \leq 22B^2\epsilon$ as shown in Subsection 4.2, we know $l^2 \leq \lambda^2 + 22B^2\epsilon$ at time $\hat{t}_0$. If the squared edge length at time t_0^{E} is also less than $\lambda^2 + 22B^2\epsilon$, given the vertex-line distance short than R, we can detect this case by setting $D^2 \geq R^2 + \frac{\lambda^2 + 22B^2\epsilon}{4}$. If not, the squared edge length at time t_0^{E} is greater than $\lambda^2 + 22B^2\epsilon$. Since t_0^{F} is a local minimum of $F_{0ij}(t)$ closest to t_0^{E} and $F_{0ij}(t_0^{\mathrm{E}}) \leq 0$, we know $F_{0ij}(t) \leq 725B^2\epsilon^{\frac{1}{2}}$, for any $t \in [t_0^{\mathrm{E}}, \hat{t}_0]$. By the definition of $F_{0ij}(t)$, the vertex-line distance is bounded by $\frac{725B^2\epsilon^{\frac{1}{2}}}{\sqrt{\lambda^2 + 22B^2\epsilon}} + S$, if $l^2 \geq \lambda^2 + 22B^2\epsilon$ at time t. By defining two events: 1) $l^2 = \lambda^2 + 22B^2\epsilon$ and 2) the vertex's projection is an endpoint of the edge, we apply the two-event analysis in Subsection 3.3.1 to derive a sufficient condition: $D^2 \geq \frac{\lambda^2 + 22B^2\epsilon}{4} + \left(\frac{725B^2\epsilon^{\frac{1}{2}}}{\sqrt{\lambda^2 + 22B^2\epsilon}} + S\right)^2$, using which vertex-vertex CCD detects the relevant collision at certain time $t \in [t_0^{\mathrm{E}}, \hat{t}_0]$.
- **Barycentric test.** After the edge length test, the barycentric test may still fail. Passing the edge length test means $\hat{l}^2 \geq \lambda^2$ at time $\hat{t}_0$, or $l^2 \geq \lambda^2 - 22B^2\epsilon$. Since barycentric weights are also calculated using dot products, their errors are bounded by $22B^2\epsilon$ and the barycentric test error is bounded by $44B^2\epsilon$. The value examined by the barycentric test is equal to the edge length times the distance of the vertex's projection to an endpoint. So when the barycentric test fails erroneously, the distance of the vertex's projection to a certain endpoint cannot exceed $\frac{(44B^2\epsilon)^2}{\sqrt{\lambda^2 - 22B^2\epsilon}}$ at time $\hat{t}_0$. Together with the upper bound on the vertex-line distance at time $\hat{t}_0$, we get a sufficient condition: $D^2 \geq \frac{(44B^2\epsilon)^2}{\lambda^2 - 22B^2\epsilon} + \left(\frac{725B^2\epsilon^{\frac{1}{2}}}{\sqrt{\lambda^2 - 22B^2\epsilon}} + S\right)^2$.



(a) Dot product (b) Computing *a* and *d* (c) *b* and *c*

(d) Computing **N** (e) Computing barycentric weights

Figure 4: *Error bounds in collision detection. The red arrows represent* `multiply` *and the green arrows represent* `add` *or* `subtract`*.*

False positives. A false positive can be detected either by vertex-edge CCD itself, or vertex-vertex CCD. Since the error in vertex-vertex CCD is small, we can consider D as an upper bound on the squared vertex-edge distance in both cases. To reduce false positives, we would like to find the optimal λ that minimizes D. After temporarily ignoring S and performing simplifications, we choose $\lambda^2 = 2080B^2\epsilon^{\frac{1}{2}}$ and we set $D^2 \geq (16B\epsilon^{\frac{1}{4}} + S)^2 + 522B^2\epsilon^{\frac{1}{2}}$ as a sufficient condition for vertex-edge CCD.

4.4 Errors in Vertex-Triangle CCD

The vertex-triangle CCD algorithm contains four steps: 1) finding the roots of a cubic equation; 2) testing the triangle area; 3) verifying the triangle heights; and 4) testing the barycentric coordinates.

Root finding. The first step is to solve $F(t) = at^3 + bt^2 + ct + d = 0$. The difference between each exact root t_0 and its computed version $\hat{t}_0$ can be caused by two reasons. The first reason is the errors associated with the coefficients. According to Theorem 4.1, $|\hat{a} - a|$ and $|\hat{d} - d|$ are bounded by $6B^3((1+\epsilon)^5 - 1)$ and $|\hat{b} - b|$ and $|\hat{c} - c|$ are bounded by $18B^3((1+\epsilon)^7 - 1)$ as Figure 4b and 4c show. The second reason is the errors caused by the cubic solver. Appendix B shows that our cubic solver guarantees the existence of $\hat{t}_0$ for every t_0 and $|F(t)| \leq \mu$ for any $t \in [t_0, \hat{t}_0]$, if $\mu \geq 33492B^3\epsilon$.

Triangle area test. In this step, we test whether the triangle area is sufficiently large at a computed collision time candidate $\hat{t}_0$. To simplify the error analysis, we test $\left\|\hat{\mathbf{N}}\right\|_\infty \geq \tau$, in which τ is an area threshold. Using Theorem 4.1 and Figure 4d, we know the error associated with $\mathbf{N}$ is bounded by $2B^2((1+\epsilon)^6 - 1) \leq 13B^2\epsilon$. If the triangle area test fails, then $A(\hat{t}_0) \leq \sqrt{3}\,\|\mathbf{N}\|_\infty \leq \sqrt{3}(\tau + 13B^2\epsilon)$ at time $\hat{t}_0$. From the analysis in Subsection 3.3, we know $R^2 \geq \frac{\tau + 13B^2\epsilon}{6} + r^2$ is sufficient to get such collisions detected, in which $r = \frac{\mu}{\tau - 13B^2\epsilon}$. On the other hand, if a vertex and a triangle passes the area test, then $\|\mathbf{N}\|_2 \geq \|\mathbf{N}\|_\infty \geq \tau - 13B^2\epsilon$. By definition, the vertex-plane distance must be bounded by r. We note that if any collision satisfies $A(t) \leq \tau - 13B^2\epsilon$ for any $t \in [t_0, \hat{t}_0]$, then $R^2 \geq \frac{\tau + 13B^2\epsilon}{6} + r^2$ is also sufficient to get it detected, according to the analysis in Subsection 3.3.1. So we assume $A(t) \geq \tau - 13B^2\epsilon$ and the vertex-plane distance is bounded by r for any $t \in [t_0, \hat{t}_0]$ next.

Triangle height test. When triangles are slim, the rounding errors in the barycentric test may cause large false positives. So we propose a triangle height test to detect them and send them to vertex-edge CCD instead. Specifically, we compare every triangle height with a height threshold η at time $\hat{t}_0$: $\eta \left\|\mathbf{x}_{ij}\right\|_\infty \leq \|\mathbf{N}\|_\infty$, in which ij is a triangle edge. Since this test happens after the area test, we must have $\|\mathbf{N}\|_2 \geq \tau - 13B^2\epsilon$. Let h be a triangle height. If it passes the height test, then $h = \frac{\|\mathbf{N}\|_2}{\|\mathbf{x}_{ij}\|_2} \geq \frac{\tau - 13B^2\epsilon}{\|\mathbf{x}_{ij}\|_2}$ and $\eta \left\|\mathbf{x}_{ij}\right\|_\infty < \|\mathbf{N}\|_\infty + 21B^2\epsilon$, in which $21B^2\epsilon$ is the error associated with this test according to Theorem 4.1 (assuming that $\eta \leq B$). So $h \geq \frac{\eta}{\sqrt{3}} - \frac{21B^2\epsilon}{\|\mathbf{x}_{ij}\|_2} \geq \frac{\eta}{\sqrt{3}} - \frac{21hB^2\epsilon}{\tau - 13B^2\epsilon}$. Since τ is going to be substantially greater than $B^2 O(\epsilon)$, it is safe to simplify the previous equation into: $2h \geq \eta$. Therefore, all of the heights must be greater than $\frac{\eta}{2}$, if a triangle passes the triangle height test. On the other hand, if a triangle fails the height test, we must ensure that a real collision still gets detected by vertex-edge CCD.

- The height test fails when any triangle edge is shorter than E at time $\hat{t}_0$. If any edge is shorter than E at time t_0, we can simply use the condition: $R^2 \geq \frac{E^2}{4}$. If not, by defining two events: 1) the shortest edge length is E and 2) the vertex's projection is on a triangle edge, and we can use the two-event analysis in Subsection 3.3.1 to derive a sufficient condition: $R^2 \geq \frac{E^2}{4} + r^2$, in which $r = \frac{\mu}{\tau - 13B^2\epsilon}$ is an upper bound on the vertex-plane distance at any time $t \in [t_0, \hat{t}_0]$.
- Otherwise, the height test fails when no edge is shorter than E at time $\hat{t}_0$. Using Theorem 4.1, we know $\eta \left\|\mathbf{x}_{ij}\right\|_\infty > \|\mathbf{N}\|_\infty - 21B^2\epsilon$. Since $\left\|\mathbf{x}_{ij}\right\|_2 \geq \left\|\mathbf{x}_{ij}\right\|_\infty$ and $\|\mathbf{N}\|_\infty \geq \frac{1}{\sqrt{3}}\|\mathbf{N}\|_2$, at least one triangle height is less than $\sqrt{3}(\eta + \frac{21B^2\epsilon}{E})$. Based on the previous analysis, we use $R^2 \geq \frac{3}{4}(\eta + \frac{21B^2\epsilon}{E})^2 + r^2$.

By temporarily ignoring η, we choose $E = \frac{21}{4}B\epsilon^{\frac{1}{2}}$, in which case the condition is reformulated into: $R^2 \geq \frac{3(\eta + 4B\epsilon^{\frac{1}{2}})^2}{4} + r^2$.

Barycentric test. The barycentric test computes the barycentric weights and compares them to 0 or $\mathbf{N} \cdot \mathbf{N}$. Each weight contains at most $12B^4((1+\epsilon)^{15} - 1) \leq 181B^4\epsilon$ error, as Figure 4e shows. So the overall error of each barycentric criterion must be bounded by $24B^4((1+\epsilon)^{16} - 1) + 12B^4((1+\epsilon)^{15} - 1) \leq 565B^4\epsilon$. If the barycentric test fails correctly, then the vertex's projection is outside of the triangle at time $\hat{t}_0$ indeed. According to our previous analysis in Subsection 3.3.1, $R \geq r$ is sufficient to get such cases detected by vertex-edge CCD. If the barycentric test fails erroneously, then the vertex's projection should still be inside of the triangle at time $\hat{t}_0$. By definition, the barycentric weight assigned to $\mathbf{x}_1$ is equal to $\|\mathbf{N}\|_2$ times the edge length $\|\mathbf{x}_{23}\|_2$ times the vertex-edge-line distance l, as Figure 5a shows. Since the vertex and the triangle passed the area test and the height test, we know $\|\mathbf{N}\|_2 \geq \tau - 13B^2\epsilon$ and all of the heights are greater than $\frac{\eta}{2}$. Given the fact that no edge can be shorter than the shortest height, we have $l \leq \frac{1130B^4\epsilon}{\eta(\tau - 13B^2\epsilon)}$. When a vertex's projection is within the triangle, its distance to the edge lines is also its distance to the edges. So $R^2 \geq \frac{(1130B^4\epsilon)^2}{\eta^2(\tau - 13B^2\epsilon)^2} + r^2$ is sufficient to get the case detected by vertex-edge CCD at $\hat{t}_0$.

4.4.1 False Positives

To study false positives, we would like to formulate an upper bound on the vertex-triangle distance at $\hat{t}_0$ when a false positive happens.

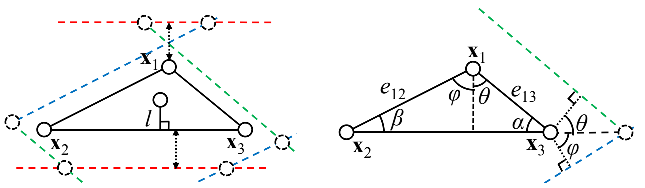
Proximity test. A false positive can occur when it erroneously passes all of the three tests. Based on the previous analysis, we know the vertex-plane distance is bounded by r at time $\hat{t}_0$. The question is what is the distance between the vertex's projection and the triangle edges, when a false positive occurs due to the barycentric test. As we showed previously, the rounding error of a barycentric criterion is bounded by $565B^4\epsilon$ and the barycentric weight is equal to $\|\mathbf{N}\|_2$ times the edge length times the vertex-edge-line distance. Let e be an Euclidean edge length at time $\hat{t}_0$. If the vertex's projection is erroneously treated as if it was inside, then its distance to the corresponding edge line must be bounded by $\frac{565B^4\epsilon}{e\|\mathbf{N}\|_2}$ at time $\hat{t}_0$. The six criteria used in the barycentric test form six lines around the triangle and a false positive can occur only if the vertex's projection is within the region surrounded by the six lines, as Figure 5a shows. So instead finding an upper bound on the distance from the vertex's projection to the triangle, we find an upper bound on the distance from the six corner points (where the lines intersect) to the triangle edges. Without loss of generality, let us consider a corner point formed by the barycentric criteria of two edges, as shown in Figure 5b. Let the two edge lengths be e_{13} and e_{12}. Since the distance from $\mathbf{x}_3$ to the two lines are $\frac{565B^4\epsilon}{e_{13}\|\mathbf{N}\|_2}$ and $\frac{565B^4\epsilon}{e_{12}\|\mathbf{N}\|_2}$, we have $e_{13}\cos\theta = e_{12}\cos\phi$, in which θ and ϕ are two angles formed by the corner point and the perpendicular directions. Let α and β be two triangle angles and we know $\alpha + \beta + \theta + \varphi = \pi$. This means the line splitting the angle at $\mathbf{x}_1$ into θ and ϕ must be perpendicular to $\mathbf{x}_2\mathbf{x}_3$. Therefore, $e_{13}\cos\theta = e_{12}\cos\varphi = h$, where h is the triangle height and we know the corner-point-vertex distance must be bounded by $\frac{565B^4\epsilon}{h\|\mathbf{N}\|_2}$. Since a false positive passes both the area test and the height test, we have $\|\mathbf{N}\|_2 \geq \tau - 13B^2\epsilon$ and $h \geq \frac{\eta}{2}$. So the corner-vertex distance must be bounded by $\frac{1130B^4\epsilon}{\eta(\tau - 13B^2\epsilon)}$, which is also an upper bound on the distance between the vertex's projection and the edges. This conclusion is valid even if α or β is greater than $\frac{\pi}{2}$, in which case θ or φ is negative. Together the squared vertex-triangle distance is bounded by $\frac{(1130B^4\epsilon)^2}{\eta^2(\tau - 13B^2\epsilon)^2} + r^2$ at time $\hat{t}_0$.

Vertex-edge test. If a false positive fails the three tests, then it passes vertex-edge CCD. According to Subsection 4.3, the squared distance is bounded by $(16B\epsilon^{\frac{1}{4}} + 3R)^2 + 522B^2\epsilon^{\frac{1}{2}}$.

In summary, our goal is to minimize:

$$\min\left(\frac{(1130B^4\epsilon)^2}{\eta^2(\tau - 13B^2\epsilon)^2} + r^2, (16B\epsilon^{\frac{1}{4}} + 3R)^2 + 522B^2\epsilon^{\frac{1}{2}}\right), \tag{2}$$

subject to $R^2 \geq \max\left(\frac{\tau + 13B^2\epsilon}{6}, \frac{3(\eta + 4B\epsilon^{\frac{1}{2}})^2}{4}, \frac{(1130B^4\epsilon)^2}{\eta^2(\tau - 13B^2\epsilon)^2}\right) + r^2$, in which $r = \frac{\mu}{\tau - 13B^2\epsilon}$. Although this minimization is complex, many terms are not so significant. After simplifications and approximations, we propose to use $\mu = 64B^3\epsilon^{\frac{3}{4}}$, $\tau = 64B^2\epsilon^{\frac{1}{2}}$, $\eta = 4B\epsilon^{\frac{1}{4}}$, and $R = 6B\epsilon^{\frac{1}{4}}$. The resulting vertex-triangle distance is approximately $41B\epsilon^{\frac{1}{4}}$. Using the double precision format, this is about 0.5 percent of the maximum edge length. We note the false positive analysis is to predict the vertex-triangle distance in the worst case and it does not need to be exact.



(a) A region formed by six dash lines parallel to the edges, in which a false positive can exist.

(b) A corner point. The distance between the corner point and $\mathbf{x}_3$ is inversely proportional to the height.

Figure 5: *False positives in vertex-triangle CCD.*

4.5 Errors in Edge-Edge CCD

Similar to vertex-triangle CCD, edge-edge CCD uses the cubic solver to compute each potential collision time t_0. The computed root $\hat{t}_0$ satisfies $|F(t)| \leq \mu$ for any $t \in [t_0, \hat{t}_0]$, if $\mu \geq 33492B^3\epsilon$. The analysis is different in the rest of the steps.

Quadrilateral area test. Similar to the triangle area test in Subsection 4.4, we use $\|\hat{\mathbf{N}}\|_\infty \geq \tau$ to test whether the quadrilateral area is sufficiently large. If this test fails, then $A(\hat{t}_0) = \|\mathbf{N}\|_2 \leq \sqrt{3}(\tau + 13B^2\epsilon)$ at time $\hat{t}_0$. From the analysis provided in Subsection 3.4, we know that $R^2 \geq \frac{\sqrt{3}(\tau+13B^2\epsilon)}{4} + r^2$ is a sufficient condition, in which $r = \frac{\mu}{\tau-13B^2\epsilon}$. Using the same analysis, we know if two edges pass the area test, we must have $\|\mathbf{N}\|_2 \geq \tau - 13B^2\epsilon$. If $A(t) \leq \tau - 13B^2\epsilon$ at any $t \in [t_0, \hat{t}_0]$, it must be detected by $R^2 \geq \frac{\sqrt{3}(\tau+13B^2\epsilon)}{4} + r^2$ in vertex-edge CCD. So we assume $A(t) \geq \tau - 13B^2\epsilon$ and the line-line distance is bounded by r at any $t \in [t_0, \hat{t}_0]$ later on.

Quadrilateral height test. Similar to the triangle height test in Subsection 4.4, the quadrilateral height test is used to reduce the false positives. For every edge $\mathbf{x}_i\mathbf{x}_j$, we define its quadrilateral height as: $\frac{\|\mathbf{N}\|_2}{\|\mathbf{x}_{ij}\|_2}$. In our test, we compare every height with a threshold η: $\eta \|\mathbf{x}_{ij}\|_\infty \leq \|\mathbf{N}\|_\infty$. Similar to the triangle height test, we know that if a height h passes the height test, then we must have $2h \geq \eta$. To ensure the detection of every real collision event, we consider two possibilities when a real collision fails this test.

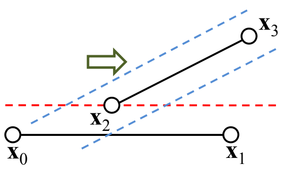
- This test may fail when any edge is shorter than E at time $\hat{t}_0$. If any edge is shorter than E at the intersection time t_0, we can simply use the condition: $R^2 \geq \frac{E^2}{4}$ to get it detected at time t_0. Otherwise, by using the same two-event analysis in Subsection 4.4, we see $R^2 \geq \frac{E^2}{4} + r^2$ is a sufficient condition.
- If the height test fails when no edge is shorter than E at time $\hat{t}_0$, at least one height is shorter than $\sqrt{3}(\eta + \frac{16B^2\epsilon}{E})$ at time $\hat{t}_0$, in which $16B^2\epsilon$ is an upper bound on the height test error. When two edges intersect, the shortest vertex-edge distance must be bounded by half of the shorter height. So we can set $R^2 \geq \frac{3}{4}(\eta + \frac{16B^2\epsilon}{E})^2 + r^2$ using the two-event analysis as before.

By using $E = 4B\epsilon^{\frac{1}{2}}$, the condition becomes: $R^2 \geq \frac{3(\eta+4B\epsilon^{\frac{1}{2}})^2}{4} + r^2$.

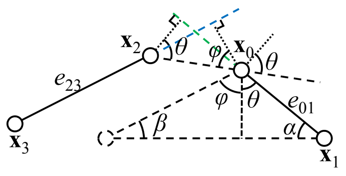
Barycentric test. Similar to the barycentric weights in vertex-triangle CCD, the error associated with each weight is bounded by $181B^4\epsilon$. So the overall error associated with each criterion must be bounded by $362B^4\epsilon$. If the barycentric test fails correctly, then the projections of the two edges do not intersect at time $\hat{t}_0$. We know from our previous analysis in Subsection 3.4 that $R \geq r$ is sufficient to get this case detected by vertex-edge CCD. If the barycentric test fails erroneously, then the projections of the two edges still intersect at time $\hat{t}_0$. By definition, the barycentric weight is equal to $\|\mathbf{N}\|_2$ times the edge length times the vertex-line distance. Since the two edges passed the area test and the height test, we know $\|\mathbf{N}\|_2$ is greater than $\tau - 13B^2\epsilon$ and the heights are greater than $\frac{\eta}{2}$. Given the fact that no edge can be shorter than the shortest height, all vertex-edge-line distances in the projected space are bounded by $\frac{724B^4\epsilon}{\eta(\tau-13B^2\epsilon)}$. Since at least one vertex-line distance is the vertex-edge distance in the projected space, at least one vertex-edge distance is bound by $\frac{724B^4\epsilon}{\eta(\tau-13B^2\epsilon)}$. So we use $R^2 \geq \frac{(724B^4\epsilon)^2}{\eta^2(\tau-13B^2\epsilon)^2} + r^2$ to detect such collisions.

4.5.1 False Positives

Next we study the upper bound on the edge-edge distance at time $\hat{t}_0$, when a false positive happens.



(a) The maximum EE distance occurs when both edge-line distances reach the upper bounds.



(b) The EE distance is inversely proportional to the height of a virtual triangle formed by aligning $\mathbf{x}_2\mathbf{x}_3$ to $\mathbf{x}_0\mathbf{x}_1$.

Figure 6: *False positives in edge-edge CCD.*

Proximity test. A false positive occurs when it passes the three tests. We know the line-line distance must still be bounded by r at $\hat{t}_0$. The question is what is the maximum distance between two projected edges at $\hat{t}_0$, when a false positive occurs? As showed previously, the error of a barycentric criterion is bounded by $362B^4\epsilon$. By definition, the weight is equal to $\|\mathbf{N}\|_2$ times the edge length times the vertex-edge-line distance. Let e be an Euclidean edge length at time $\hat{t}_0$. If the vertex's projection is erroneously treated as if it was on the wrong side, then its distance to the other edge line must be bounded by $\frac{362B^4\epsilon}{e\|\mathbf{N}\|_2}$ at time $\hat{t}_0$. Let $\mathbf{x}_0\mathbf{x}_1$ and $\mathbf{x}_2\mathbf{x}_3$ be two edges with fixed lengths and orientations. The maximum edge-edge distance happens if and only if the distance from the edge $\mathbf{x}_0\mathbf{x}_1$ to the line $\mathbf{x}_2\mathbf{x}_3$ is $\frac{362B^4\epsilon}{e_{23}\|\mathbf{N}\|_2}$, and the distance from the edge $\mathbf{x}_2\mathbf{x}_3$ to the line $\mathbf{x}_0\mathbf{x}_1$ is $\frac{362B^4\epsilon}{e_{01}\|\mathbf{N}\|_2}$, in which e_{01} and e_{23} are the edge lengths. Because if not, we can move the two edges to increase the edge-edge distance, such as moving $\mathbf{x}_2\mathbf{x}_3$ to the right side as Figure 6a shows. Since these two edges do not intersect, there is no quadrilateral in this configuration. Instead, we can translate $\mathbf{x}_0\mathbf{x}_1$ to form a triangle as Figure 6b shows. Similar to the analysis in Subsection 4.4.1, we have $\alpha + \beta + \theta + \varphi = \pi$ and the triangle height splits the angle at $\mathbf{x}_2$ into θ and ϕ. So $e_{01}\cos\theta = e_{23}\cos\varphi = H$, in which H is the triangle height. Since the new edge cannot be longer than $2e_{01}$ nor $2e_{23}$, H must be greater than half of the shorter triangle height corresponding to e_{01} or e_{23}, which is also a quadrilateral height. Therefore, $H \geq \frac{h}{2} \geq \frac{\eta}{4}$. Since a false positive passes the area test and the height test, we bound the edge-edge distance in the projected space by $\frac{1448B^4\epsilon}{\eta(\tau-13B^2\epsilon)}$, according to Figure 6b. Together, the squared edge-edge distance must be bounded by: $\frac{(1448B^4\epsilon)^2}{\eta^2(\tau-13B^2\epsilon)^2} + r^2$.

Vertex-edge test. If a false positive fails any of the three test, it can pass vertex-edge CCD only. The analysis in Subsection 4.3 shows the squared distance is bounded by $(16B\epsilon^{\frac{1}{4}}+3R)^2+522B^2\epsilon^{\frac{1}{2}}$.

In summary, we have:

$$\min\left(\frac{(1448B^4\epsilon)^2}{\eta^2(\tau-13B^2\epsilon)^2} + r^2, (16B\epsilon^{\frac{1}{4}} + 3R)^2 + 522B^2\epsilon^{\frac{1}{2}}\right), \tag{3}$$

subject to $R^2 \geq \max\left(\frac{\sqrt{3}(\tau+13B^2\epsilon)}{4}, \frac{3(\eta+4B\epsilon^{\frac{1}{2}})^2}{4}, \frac{(724B^4\epsilon)^2}{\eta^2(\tau-13B^2\epsilon)^2}\right) + r^2$. After simplifications and approximations, we propose to use $\mu = 64B^3\epsilon^{\frac{3}{4}}$, $\tau = 64B^2\epsilon^{\frac{1}{2}}$, $\eta = 4B\epsilon^{\frac{1}{4}}$, and $R = 6B\epsilon^{\frac{1}{4}}$. The resulting edge-edge distance is approximately $41B\epsilon^{\frac{1}{4}}$. Using the double precision format, this is about 0.5 percent of the maximum edge length.

5 Results

Based on an existing cubic-based CCD system, we implemented our modifications in less than an hour. We also implemented plane-based culling (not including AABB culling) in our CCD tests, as proposed by Brochu and Bridson [2012]. Instead of using interval arithmetic, we avoid the influence of rounding errors on plane-

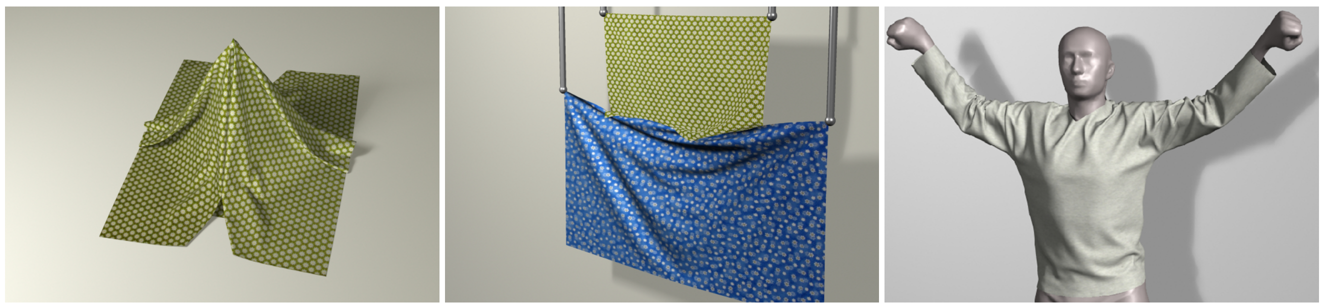


Figure 7: *Animation examples. These are the examples used to test the performance and robustness of our system.*

based culling by introducing another error tolerance. To further speed up our algorithms, we developed another culling method for the vertex-edge test. Specifically, given the computed collision time candidate $\hat{t}_0$, we first test whether the magnitude of the cubic function value goes beyond μ at both $\hat{t}_0 \pm \epsilon^{\frac{1}{2}}$. If so, when collision happens at t_0, we must have $|\hat{t}_0 - t_0| \leq \epsilon^{\frac{1}{2}}$. Given B as an upper bound on the relative velocities, the distance between the vertex's projection and the triangle must be bounded by $\sqrt{3}B\epsilon^{\frac{1}{2}}$ at time $\hat{t}_0$, and at least one barycentric weight must be in $[-9B^4\epsilon^{\frac{1}{2}}, \|\mathbf{N}\|_2^2 + 9B^4\epsilon^{\frac{1}{2}}]$. In other words, no collision can happen if the weight is beyond this range. Since ϵ is small, this test can eliminate more than 99.9 percent of VE tests as our experiment shows.

One issue we have not considered yet is the errors in vector computation. Take vertex-vertex CCD for example. The goal is to find the collision between $\{\mathbf{x}_i^0, \mathbf{x}_i^1\}$ and $\{\mathbf{x}_j^0, \mathbf{x}_j^1\}$, but the algorithm tests $\{\mathbf{x}_i^0, \mathbf{x}_i^1\}$ and $\{\mathbf{x}_i^0 + \hat{\mathbf{x}}_{ji}, \mathbf{x}_i^1 + \hat{\mathbf{x}}_{ji} + \hat{\mathbf{v}}_{ji}\}$. Since such errors are bounded by $BO(\epsilon)$, if a collision failure happens because of this, the vertex-triangle distance at the ending time must be bounded by $BO(\epsilon)$. This problem can be solved by doing proximity tests at the ending time as in many existing systems, including ours. Since our tolerances are larger, they guarantee the detection of such collisions.

Another issue comes from the upper bound B, when it is computed for every time step. Let B_0 be the upper bound in the previous time step and B_1 be the upper bound in the next time step. If $B_1 > B_0$, collisions may be detected at the starting time of the next step, causing convergence issues in collision handling. One possible solution is to go back to the previous time step and use B_1 for collision detection at the ending time. For simplicity, we predefine B and assume that it is valid through the whole animation. We can even define B separately for every vertex-triangle or edge-edge pair, using their resting shapes. Doing this reduces B, when handling adaptively sampled meshes with both long and short edges.

Performance. We first test the performance of our algorithms on four cloth animation examples as shown in Figure 1 and 7, using a single core of a 2.67GHz Intel Xeon X5650 processor. After AABB culling, the average computational time of each basic vertex-triangle or edge-edge CCD test is 350ns, while the computational time after using our modifications is 355ns. So the overhead is approximately 1.4 percent of the basic cost. Without plane-based culling, the costs before and after our modifications are 409ns and 416ns. To better understand the performance in complex situations, we construct a synthetic example by randomly sampling vertex positions and velocities in a unit cube and we exhaustively tested 10 billion cases (taking about one hour to run). The average computational costs per test before and after our modifications are 364ns and 369ns. The timing difference between the synthetic example and the animation examples is due to more positive cases.

Compared with the exact approach proposed by Brochu and Bridson [2012], our tests run 20 to 40 times faster than exact tests after collision culling. This is highly noticeable in the synthetic example, where nearly half cases pass AABB culling and nearly one third cases pass plane-based culling. Our experiment shows the average time per exact test is 4,962ns after AABB culling. In contrast, our tests are fast even without culling. In real animation examples, however, such difference may not be so noticeable because of collision culling. The only difference we found occurs at the collision peaking time, when our approach can be up to 20 percent faster than the exact approach. We believe the efficiency of our approach will be more obvious, once the cost of collision culling becomes less dominant in the overall collision handling cost.

False positives. According to our previous analysis, the maximum vertex-triangle (or edge-edge) distance of a false positive is approximately 20μm in our simulation. To evaluate the influence of false positives on animation examples, we cannot simply count them among all of the positive cases, because a false positive may just cause earlier collision handling and avoid true positives from happening later. So instead, we count the total numbers of tested cases and their positive results, when we simulate the animation examples using our approach and the exact approach respectively. Our experiment shows that the total case numbers are about the same when using the two approaches, but the system can detect up to 10 percent more positive cases when using our approach. Fortunately, since most results are negative, these additional positives have little influence on the collision detection performance.

Safety. Collision detection failures can be problematic in cubic-based CCD tests, if no error tolerance is used. Even when tolerances are used, it is still possible to construct failure cases based on the two degenerate cases as shown in Subsection 3.3. Interestingly, it is extremely rare to see failures in cloth animation and the colliding cloth example in Figure 7b is the only failure one we found. In this example, basic CCD tests missed multiple collisions at first. Shortly after that, a burst of collisions were detected and collision handling failed to converge. We did not apply other techniques (such as impact zones) to fix it after this. We believe that collision failures are rare in cloth animation due to two main reasons. The first reason is because physically based simulation often avoids small triangle or short edge cases for numerical stability. If we assume that the triangle areas and the edge lengths remain approximately constant, then we can reduce the error bounds substantially. The second reason is that the failure of a single case does not necessarily cause the failure of the whole collision handling process. For instance, ill-shaped triangles or edges may be surrounded by good-shaped ones, whose correct test results will prevent failures from happening. It will be interesting to see whether basic CCD tests are more likely to fail in other cases, such as hair animation.

Strengths and limitations. Given the fact that it is rare to see collision failures in basic CCD algorithms, the question is: *why is it still necessary to use our algorithms?* This is due to the uncertainty of collision detection failures. No matter how large the error tolerances we use and how rare collision detection failures are, the risk still exists in basic CCD algorithms. This risk can be reduced by increasing error tolerance values, but that will invite more false positives. In contrast, our method not only eliminates collision failures, but also reduces false positives in an automatic way. Our method is also affordable and easy to implement.

Due to a number of simplifications we made in the analysis, the error bounds and the suggested tolerance values are not the tightest. However, it is unlikely to make them asymptotically smaller, without substantially modifying our approach. The conditions and the modifications we proposed in this paper are sufficient, and it is not clear whether they are always necessary. In our analysis, we assume that the errors are independent and we formulate the bounds and the conditions in the worst scenario. In reality, some errors may be related and they may not be large simultaneously.

6 Conclusion and Future Work

We proposed a simple yet effective approach to eliminate collision detection failures in cubic-based CCD algorithms. Based on our forward error analysis, we draw two additional conclusions. 1) It is a good idea to use double precision rather than single precision. When using the single-precision floating-point format in our method, the error caused by false positives can be as large as half of the maximum edge length. 2) Large time steps not only cause numerical instability in simulation, but also increase false positives, since the vector bound B becomes larger.

In the future, we are interested in understanding the errors when machine epsilon varies, i.e., under the extended floating-point format. Some computational steps, such as normal computation, are more important than the others to our approach. We would like to analyze whether it is worthwhile to compute them by exact arithmetic. Finally, we plan to study the compatibility and performance of our method on GPU and with floating-point optimizations.

Acknowledgments

The author would like to thank Nvidia and Adobe for their funding and equipment support.

References

AINSLEY, S., VOUGA, E., GRINSPUN, E., AND TAMSTORF, R. 2012. Speculative parallel asynchronous contact mechanics. *ACM Transactions on Graphics (SIGGRAPH Asia 2012) 31*, 6 (Nov.), 151:1.

BARAFF, D., WITKIN, A., AND KASS, M. 2003. Untangling cloth. *ACM Transactions on Graphics (SIGGRAPH) 22* (July), 862–870.

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. In *Proc. of ACM SIGGRAPH 2002*, E. Fiume, Ed., vol. 21 of *Computer Graphics Proceedings, Annual Conference Series*, 594–603.

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

GILBERT, E. G., JOHNSON, D. W., AND KEERTHI, S. S. 1988. A fast procedure for computing the distance between complex objects in three-dimensional space. *Robotics and Automation 4*, 2, 193–.

GOLDBERG, D. 1991. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys 23*, 1.

HARMON, D., VOUGA, E., TAMSTORF, R., AND GRINSPUN, E. 2008. Robust treatment of simultaneous collisions. *ACM Transactions on Graphics (SIGGRAPH) 27*, 3 (August), 23:1–23:4.

HARMON, D., VOUGA, E., SMITH, B., TAMSTORF, R., AND GRINSPUN, E. 2009. Asynchronous contact mechanics. *ACM Transactions on Graphics (SIGGRAPH) 28*, 3 (July), 87:1–87:12.

HARMON, D., ZHOU, Q., AND ZORIN, D. 2011. Asynchronous integration with phantom meshes. In *Proc. of SCA*, 247–256.

HUH, S., METAXAS, D. N., AND BADLER, N. I. 2001. Collision resolutions in cloth simulation. In *Proc. of Computer Animation*, IEEE, 122–127.

LARSEN, E., GOTTSCHALK, S., LIN, M. C., AND MANOCHA, D. 2000. Fast distance queries with rectangular swept sphere volumes. In *Proc. of Robotics and Automation*, 3719–3726.

LAUTERBACH, C., MO, Q., AND MANOCHA, D. 2010. gProximity: Hierarchical GPU-based operations for collision and distance queries. In *Proc. of Eurographics*, vol. 29, 419–428.

PABST, S., KOCH, A., AND STRAßER, W. 2010. Fast and scalable CPU/GPU collision detection for rigid and deformable surfaces. *Computer Graphics Forum 29*, 5, 1605–1612.

PROVOT, X. 1997. Collision and self-collision handling in cloth model dedicated to design garments. In *Computer Animation and Simulation*, 177–189.

SCHVARTZMAN, S. C., PÉREZ, A. G., AND OTADUY, M. A. 2010. Star-contours for efficient hierarchical self-collision detection. *ACM Transactions on Graphics (SIGGRAPH 2010) 29* (July), 80:1.

SELLE, A., SU, J., IRVING, G., AND FEDKIW, R. 2009. Robust high-resolution cloth using parallelism, history-based collisions, and accurate friction. *IEEE Transactions on Visualization and Computer Graphics 15* (March), 339–350.

SHEWCHUK, J. R. 1996. Adaptive precision floating-point arithmetic and fast robust geometric predicates. *Discrete & Computational Geometry 18*, 305–363.

STAM, J. 2009. Nucleus: Towards a unified dynamics solver for computer graphics. In *11th IEEE International Conference on Computer-Aided Design and Computer Graphics*.

TANG, M., KIM, Y. J., AND MANOCHA, D. 2010. Continuous collision detection for non-rigid contact computations using local advancement. In *ICRA*, 4016–4021.

TANG, M., MANOCHA, D., YOON, S.-E., DU, P., HEO, J.-P., AND TONG, R.-F. 2011. VolCCD: Fast continuous collision culling between deforming volume meshes. *ACM Transactions on Graphics 30*, 5 (Oct.), 111:1–111:15.

THOMASZEWSKI, B., PABST, S., AND STRAßER, W. 2008. Asynchronous cloth simulation. In *Proc. of Computer Graphics International*.

VOLINO, P., AND MAGNENAT-THALMANN, N. 2006. Resolving surface collisions through intersection contour minimization. *ACM Transactions on Graphics (SIGGRAPH) 25* (July), 1154–1159.

WILKINSON, J. H. 1994. *Rounding Errors in Algebraic Processes*. Dover Publications, Incorporated.

ZHENG, C., AND JAMES, D. L. 2012. Energy-based self-collision culling for arbitrary mesh deformations. *ACM Transactions on Graphics (SIGGRAPH 2012) 31*, 4 (July), 98:1–98:12.