Floating-Point Robustness in Parametric Surface Continuous Collision Detection: From Algorithm to Benchmarking

XUWEN CHEN*, School of Intelligence Science and Technology, Peking University, China
JUNYU WANG*, Southeast University, China
CHENG YU, School of Intelligence Science and Technology, Peking University, China
XINGYU NI, University of Hong Kong, Hong Kong SAR
MENG ZHANG, School of Computer Science and Engineering, Nanjing University of Science and Technology, China
BIN WANG, Independent Researcher, China
MENGYU CHU†, State Key Laboratory of General Artificial Intelligence, Peking University, China
BAOQUAN CHEN†, Peking University, China

Continuous Collision Detection is essential in simulation and modeling for accurately identifying object collisions. While robust CCD techniques have matured for triangle meshes, ensuring floating-point robustness for parametric surfaces remains an open challenge due to their representational complexity and heightened algorithmic sensitivity. In this paper, we present the first floating-point-robust CCD framework for parametric surfaces. Built on the Time-Dependent Inclusion-Based Method (TDIBM), our approach introduces a novel error decomposition strategy that separates coefficient and arithmetic errors, enabling structured analysis and safety guarantees. To rigorously benchmark robustness, we develop a rational-arithmetic-based dataset by inverting the CCD process: we generate exact ground-truth datasets from prescribed collision outcomes. Our construction captures both typical scenarios and near-degenerate cases. We evaluate several CCD algorithms using this benchmark to provide an in-depth analysis. Together, our method and dataset establish a comprehensive foundation for analyzing, benchmarking, and improving floating-point robustness in parametric surface CCD. Code and dataset will be published upon acceptance.

CCS Concepts: • **Computing methodologies** → **Physical simulation**; • **Applied computing** → *Physics*.

Additional Key Words and Phrases: parametric surface, continuous collision detection, error analysis, dataset

ACM Reference Format:
Xuwen Chen, Junyu Wang, Cheng Yu, Xingyu Ni, Meng Zhang, Bin Wang, Mengyu Chu, and Baoquan Chen. 2026. Floating-Point Robustness in Parametric Surface Continuous Collision Detection: From Algorithm to Benchmarking. *ACM Trans. Graph.* 45, 4, Article 141 (July 2026), 14 pages. https://doi.org/10.1145/3811310

*Joint first authors; † Co-corresponding authors.

Authors' addresses: Xuwen Chen, pku_xwchen@163.com, School of Intelligence Science and Technology, Peking University, Beijing, China; Junyu Wang, juggjunyu@gmail.com, Southeast University, Nanjing, China; Cheng Yu, chengyupku@163.com, School of Intelligence Science and Technology, Peking University, Beijing, China; Xingyu Ni, xingyuni.cs@gmail.com, University of Hong Kong, Hong Kong, Hong Kong SAR; Meng Zhang, lynnzephyr@gmail.com, School of Computer Science and Engineering, Nanjing University of Science and Technology, China; Bin Wang, binwangbuaa@gmail.com, Independent Researcher, Beijing, China; Mengyu Chu, mchu@pku.edu.cn, State Key Laboratory of General Artificial Intelligence, Peking University, China; Baoquan Chen, baoquan@pku.edu.cn, Peking University, Beijing, China.

ACM 0730-0301/2026/7-ART141
https://doi.org/10.1145/3811310

1 INTRODUCTION

In simulations, Continuous Collision Detection (CCD) plays a crucial role in ensuring accurate interactions between objects by identifying the earliest time and location of collision between two objects within a single time step. A missed collision (a *false negative, FN*) can lead to interpenetration and persistent locking issues, while a spurious collision (a *false positive, FP*) typically impacts performance without compromising physical correctness. Error analysis is critical in this context: small perturbations in input or computation can flip branching decisions and yield fundamentally different algorithmic outcomes—including FNs and FPs. Ensuring robustness under error perturbations has long been a central goal in computational geometry [Attene 2020; Shewchuk 1996; Yap 2004] and CAD [Feito et al. 2013; Keyser et al. 2002].

While robust CCD has been extensively studied for triangle meshes [Brochu et al. 2012; Tang et al. 2014; Wang 2014; Wang et al. 2015], provably safe CCD for parametric surfaces remains unaddressed, due to two key challenges: (1) **Representational complexity:** Unlike triangle meshes, parametric surfaces lack closed-form solutions for collision queries. Their nonlinear behavior yields a wider variety of intricate contact scenarios, making it difficult to symbolically compute ground truth or construct exhaustive test cases. (2) **Algorithmic sensitivity**: state-of-the-art CCD methods for parametric surfaces rely on sophisticated numerical procedures that are highly sensitive to errors, undermining robustness and complicating error analysis.

We focus on tensor-product Bézier patches and propose solutions to both challenges. To improve **algorithmic robustness**, we develop a floating-point-robust CCD method based on the state-of-the-art time-dependent inclusion-based method (TDIBM) [Chen et al. 2024]. Central to our approach is a novel error analysis technique tailored to the structure of TDIBM. Due to the high sensitivity of parametric surface CCD to floating-point errors—especially through intermediate branching and geometric operations—traditional cumulative error bounds are ineffective. We resolve this by introducing a decomposition of numerical errors into coefficient errors and arithmetic errors, enabling precise reasoning about the origins and propagation of inaccuracies. We further present practical countermeasures that improve the robustness of CCD computations under these conditions.

Targeting the **representational complexity**, specifically the lack of analytic solutions and difficulty in constructing corner-case scenarios, we introduce a construction method for ground-truth CCD datasets across a broad spectrum of collision scenarios. Our key insight is to invert the CCD process. We begin with prescribed collision outcomes and solve for valid parametric surface configurations (control point positions and velocities) under randomized construction while discarding infeasible cases. Leveraging exact rational arithmetic, the resulting dataset is free of numerical artifacts. It encompasses both randomly generated general scenarios that are more commonly encountered in simulations and challenging corner cases prone to false results, enabling rigorous evaluation across a wide spectrum of parametric collision scenarios.

Our contributions are summarized as follows:

- A floating-point-robust CCD algorithm based on TDIBM, with proven safety under errors.
- A dataset generation pipeline that yields rational-arithmetic-based CCD examples with ground-truth exact in the rational number domain.
- Evaluation of existing parametric surface CCD methods on this dataset, revealing key insights into their robustness and efficiency.

2 RELATED-WORK

CCD for parametric surfaces. CCD problems between parametric surfaces can be broadly categorized into four categories. *Linearization-based methods* [Buchenau and Guthe 2021; Ferguson et al. 2023; Suwelack et al. 2013; Xiong et al. 2023] approximate parametric surfaces with piecewise linear triangular patches, reducing the problem to triangle-triangle CCD where mature algorithms are readily available [Tang et al. 2014; Wang et al. 2022, 2021]. *Sampling-based methods* [Galligo and Pavone 2006; Lu 2011; Lu and Zheng 2014; Zhang et al. 2023b] sample discrete points on one surface and test for collisions with the other. *Sum-of-squares programming* (SOSP) [Marschner et al. 2021; Zhang et al. 2023a] relaxes collision constraints into polynomial inequalities of even degree and then solve them as semidefinite programming (SDP) problems. *Inclusion-based methods* [Chen et al. 2024; Snyder 1992; Snyder et al. 1993; Von Herzen et al. 1990] use interval arithmetic to analyze potential intersections and apply a branch-and-bound scheme to recursively refine collision region until convergence. Our work builds upon the method proposed by Chen et al. [2024], currently the only approach that achieves acceptable computational performance across general CCD scenarios involving nonlinear parametric surfaces.

Safe CCD against floating-point errors. Safe CCD between triangle meshes follows two main strategies. The first category calculates theoretically exact Boolean answer to the CCD question [Brochu et al. 2012; Tang et al. 2014]. Restricted to basic arithmetic operations (such as addition, subtraction, multiplication), these methods yield exact predicate evaluations free from both FPs and FNs using Exact Geometric Computation (EGC) [Yap 2004]. They avoid complex constructions in variables involved in geometric predicates and can not give results of collision time and location. Others analyze floating-point errors throughout the CCD algorithm and derive upper bounds on accumulated errors [Wang 2014; Wang et al. 2015]. During positive/negative (p/n) decisions, if values fall below the upper bound of floating-point uncertainty, the algorithm conservatively reports a positive outcome. This strategy guarantees the absence of FNs at the cost of increasing FP numbers. Due to the complexity of parametric surfaces, designing exact CCD algorithms is impractical; thus, we adopt an error-bound approach.

CCD dataset. To evaluate CCD methods, Wang et al. [2021] proposed a large-scale benchmark for linear triangle patches, including both a handcrafted dataset exhibiting geometric and numerical degeneracies, and a simulation dataset abstracted from simulation practice. The dataset design draws inspiration partly from the fundamental geometric collision scenarios described by Erleben [2018]. However, extending these scenarios to parametric patches with increased degrees of freedom and flexibility is challenging. Additionally, Wang et al. [2021] computed the ground truth using Mathematica [Wolfram Research, Inc. 2024], taking several seconds per case. However, in the case of parametric surface CCD, the collision constraints may admit infinitely many solutions due to non-isolated contacts (e.g., along a common curve), causing Mathematica to take more than a day for a single instance. This renders such an approach infeasible for generating a sufficient number of ground-truth data.

3 BACKGROUND

This section reviews the TDIBM pipeline solving CCD problems between parametric surfaces.

Problem Definition. The CCD between two parametric surfaces $\boldsymbol{S}_1(u_1, v_1, t)$ and $\boldsymbol{S}_2(u_2, v_2, t) \subset \mathbb{R}^3$ is a constrained optimization:

$$\min_{t,u_1,v_1,u_2,v_2} \quad t \quad \text{s.t.} \quad \begin{array}{l} \boldsymbol{S}_1(u_1, v_1, t) = \boldsymbol{S}_2(u_2, v_2, t) \\ 0 \le u_1, v_1, u_2, v_2 \le 1 \\ 0 \le t \le \Delta T \end{array} \tag{1}$$

where (u_1, v_1), (u_2, v_2) are the parametric coordinates and t represents the time. The solution is the earliest collision moment with the corresponding parametric coordinates of the contact. No solution means collision-free. Considering Bézier surfaces of order $n \times m$, TDIBM assumes that control points move linearly and the surface lies within their convex hull:

$$\boldsymbol{S}(u, v, t) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_i^n(u) B_j^m(v) (\boldsymbol{p}_{ij} + t\dot{\boldsymbol{p}}_{ij}). \tag{2}$$

$B_i^n(u)$, $B_j^m(v)$ are the Bernstein polynomial basis, and $\boldsymbol{p}_{ij}, \dot{\boldsymbol{p}}_{ij}$ represent the positions and velocities of the control points.

Overall Pipeline. TDIBM addresses Eq. (1) by recursively narrowing the parametric domain from $I_{u_1} \times I_{v_1} \times I_{u_2} \times I_{v_2} = [0, 1]^4$ to a sufficiently small subdomain. In each iteration, it examines a pair of sub-surfaces with the parametric range of $[u_1^{\mathrm{L}}, u_1^{\mathrm{U}}] \times [v_1^{\mathrm{L}}, v_1^{\mathrm{U}}] \times [u_2^{\mathrm{L}}, u_2^{\mathrm{U}}] \times [v_2^{\mathrm{L}}, v_2^{\mathrm{U}}]$ and a time interval $[t^{\mathrm{L}}, t^{\mathrm{U}}]$. It identifies a potential collision window $[t_{\mathrm{sub}}^{\mathrm{L}}, t_{\mathrm{sub}}^{\mathrm{U}}] \subset [t^{\mathrm{L}}, t^{\mathrm{U}}]$ where the oriented bounding boxes (OBBs) of the surfaces intersect, i.e.:

$$\max_{ij}[(\boldsymbol{p}_{ij} + t\dot{\boldsymbol{p}}_{ij}) \cdot \boldsymbol{l}_r] \le \min_{ij}[(\boldsymbol{q}_{ij} + t\dot{\boldsymbol{q}}_{ij}) \cdot \boldsymbol{l}_r], \text{AND} \tag{3a}$$

$$\max_{ij}[(\boldsymbol{q}_{ij} + t\dot{\boldsymbol{q}}_{ij}) \cdot \boldsymbol{l}_r] \le \min_{ij}[(\boldsymbol{p}_{ij} + t\dot{\boldsymbol{p}}_{ij}) \cdot \boldsymbol{l}_r]. \tag{3b}$$

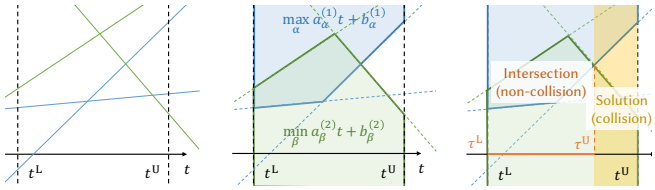

Fig. 1. Illustration of Eq.(4) and its solution. The region where the max-envelope and min-envelope do not intersect defines the interval where the inequality holds—i.e., the potential collision window.

$\{\boldsymbol{l}_r\}_{1\le r\le 15}$ are the 15 separating axis, and $\boldsymbol{p}_{ij}, \boldsymbol{q}_{ij}$ are control points of sub-surfaces. If a feasible solution $[t_{\mathrm{sub}}^{\mathrm{L}}, t_{\mathrm{sub}}^{\mathrm{U}}]$ exists, the algorithm subdivides each parametric interval, pairs them with $[t_{\mathrm{sub}}^{\mathrm{L}}, t_{\mathrm{sub}}^{\mathrm{U}}]$, and inserts them into a priority queue sorted by ascending $t_{\mathrm{sub}}^{\mathrm{L}}$. Otherwise, the subdomain is discarded as collision-free. This process recurs until either a sub-surface pair satisfies a convergence criterion (e.g., all parametric intervals have been refined below a user-specified threshold, such as 10^{-6}, indicating that a collision has been detected) or all intervals are discarded (no collision).

Solving Eq. (3). The essential step of Eq. (3) involves evaluating 30 inequalities of the form of

$$\max_{0\le\alpha\le n_1 m_1} (a_\alpha^{(1)} t + b_\alpha^{(1)}) \ge \min_{0\le\beta\le n_2 m_2} (a_\beta^{(2)} t + b_\beta^{(2)}). \tag{4}$$

The final result is obtained by intersecting feasible intervals from all inequalities. Each inequality compares the maximum of a linear-function set (with respect to t) to the minimum of another set, defining piecewise-linear concave and convex envelopes, respectively. We construct these envelopes efficiently by sorting line slopes and tracking active segments, as illustrated in Fig. 1. Because a concave envelope can only exceed a convex one over one connected interval (the infeasible region $[\tau_r^{\mathrm{L}}, \tau_r^{\mathrm{U}}]$), we can compute the feasible region by sweeping and removing that intersection from the candidate interval. The solution $[t_{\mathrm{sub}}^{\mathrm{L}}, t_{\mathrm{sub}}^{\mathrm{U}}]$ covers all the remaining intervals.

Detailed Math Steps. Solving Eq. (4) includes the following steps:

- **Step I:** Computing $a_\alpha^{(1)}, b_\alpha^{(1)}, a_\beta^{(2)}, b_\beta^{(2)}$;
- **Step II:** Constructing the envelopes. Taking the max-envelope as an example, after sorting the lines by ascending slopes $\{a_\alpha^{(1)}\}$ and discarding lines with duplicate slopes but smaller intercepts, the algorithm iteratively processes each line while maintaining a stack (indexed from 0) of selected envelope segments. In each iteration, the new line $(a_\alpha^{(1)} t + b_\alpha^{(1)})$ replaces previously stacked ones when $\Phi < 0$ (i.e., it intersects earlier and dominates), and is added to the stack when $\phi < 0$, where

$$\Phi = \begin{cases} (a_\alpha^{(1)} - a_{\gamma-1}^{(1)})(b_\gamma^{(1)} - b_{\gamma-1}^{(1)}) - (a_\gamma^{(1)} - a_{\gamma-1}^{(1)})(b_\alpha^{(1)} - b_{\gamma-1}^{(1)}), & \Gamma > 0, \text{ (5a)} \\ (a_\gamma^{(1)} - a_\alpha^{(1)}) t^{\mathrm{L}} + (b_\gamma^{(1)} - b_\alpha^{(1)}), & \Gamma = 0, \text{ (5b)} \end{cases}$$

$$\phi = (a_\gamma^{(1)} - a_\alpha^{(1)}) t^{\mathrm{U}} + (b_\gamma^{(1)} - b_\alpha^{(1)}). \tag{6}$$

Γ is the stack size. γ and $\gamma - 1$ index the top 2 lines on the stack.

- **Step III:** Determining the infeasible interval where max- and min-envelopes intersect. The algorithm first checks if the max-envelope starts below the min one using

$$\psi = (a_0^{(1)} - a_0^{(2)}) t^{\mathrm{L}} + (b_0^{(1)} - b_0^{(2)}). \tag{7}$$

If $\psi < 0$, the left endpoint is set as $\tau^{\mathrm{L}} = t^{\mathrm{L}}$. Otherwise, the algorithm traverses both envelopes using indices α and β, checking for segment intersection when both $\Psi_{\max} < 0$ and $\Psi_{\min} < 0$, with

$$\Psi_{\max} = \begin{cases} (a_{\alpha+1}^{(1)} - a_\beta^{(2)})(b_\alpha^{(1)} - b_\beta^{(2)}) - (a_\alpha^{(1)} - a_\beta^{(2)})(b_{\alpha+1}^{(1)} - b_\beta^{(2)}), \alpha < \Gamma^{(1)}\text{-}1, \text{ (8a)} \\ (a_\alpha^{(1)} - a_\beta^{(2)}) t^{\mathrm{U}} + (b_\alpha^{(1)} - b_\beta^{(2)}), \quad \alpha = \Gamma^{(1)}\text{-}1. \text{ (8b)} \end{cases}$$

$$\Psi_{\min} = \begin{cases} (a_\alpha^{(1)} - a_{\beta+1}^{(2)})(b_\alpha^{(1)} - b_{\beta+1}^{(2)}) - (a_\alpha^{(1)} - a_\beta^{(2)})(b_\alpha^{(1)} - b_{\beta+1}^{(2)}), \beta < \Gamma^{(2)}\text{-}1, \text{ (9a)} \\ (a_\alpha^{(1)} - a_\beta^{(2)}) t^{\mathrm{U}} + (b_\alpha^{(1)} - b_\beta^{(2)}), \quad \beta = \Gamma^{(2)}\text{-}1. \text{ (9b)} \end{cases}$$

Once found, the left endpoint is set as

$$\tau^{\mathrm{L}} = -(a_\alpha^{(1)} - a_\beta^{(2)})^{-1} (b_\alpha^{(1)} - b_\beta^{(2)}). \tag{10}$$

If not, the traversal terminates when $c_\beta <= a_\alpha$. A reverse pass can be used to compute the right endpoint.

4 ERROR BOUND ANALYSIS

4.1 Where Do Errors Come From?

Theoretically, due to the conservativeness established in the convex-hull assumption, TDIBM may produce FP but guarantees no FN if exact arithmetic is used. However, floating-point computations can introduce errors that lead to FN. In particular, Step I performs floating-point computations whose errors accumulate in a relatively analyzable manner. Steps II and III then heavily rely on these inexact quantities to make a large number of branching decisions based on floating-point comparisons. Once any of these decisions is corrupted, the error may alter not only the numerical values but also the topology of the constructed envelopes, potentially resulting in entirely incorrect or even non-polygonal geometries. For example, flaws in sorting and line selection can trigger such topological failures, while an inexact τ^{L} may cause incorrect merging of intervals.

This behavior fundamentally differs from triangle-based CCD, whose collision tests admit closed-form predicates and thus allow conservative evaluation by enlarging the decision thresholds with derived error bounds. In contrast, the numerous comparison-driven decisions in our setting have ambiguous and non-monotonic contributions to the final result. Such topological disruptions lie beyond the scope of conventional error bound analysis [Wang 2014].

4.2 Core Idea for Robust Error Handling

The key idea of our error analysis and mitigation framework is to decouple error sources that are otherwise intractable to analyze jointly. In Step II, we introduce an error-source separation strategy that independently estimates errors arising from coefficient computation and geometric construction. Specifically, we analyze (1) the error induced by inexact coefficients assuming an exact convex hull, and (2) the error induced by an inexact convex hull assuming exact coefficients. This separation enables us to conservatively bound the total error even in the presence of potential topological failures.

The accumulated errors in Step III make analysis through conventional methods particularly difficult. We therefore abandon the error accumulation which acts as a posterior error correction and instead introduce a preprocessing scheme, which ensures a conservative result by intercepting earlier errors and tolerating their deviations. We first review floating-point error properties and then elaborate on our new analysis method.

4.3 Properties of Floating-Point Error

According to IEEE 745 Standard, a number a should be correctly rounded to a floating-point number $\hat{a}$ that satisfies $|a - \hat{a}| < |a|\epsilon$. The machine epsilon ϵ is 2^{-17} in single-precision floating-point representation and 2^{-52} in double-precision floating-point representation. Similar rules apply to the four basic arithmetic operations (addition, subtraction, multiplication and division). When acting on two arbitrary numbers a and b, the result calculated by an exact operator $\otimes$ and that by its floating-point operator $\hat{\otimes}$ should satisfy the constraint of $|a\hat{\otimes}b - a \otimes b| < |a \otimes b|\epsilon$. Here we use a hat over a variable or an operator to denote its floating-point correspondence. Based on this, Wang [2014] proposed the following theorem:

Theorem 4.1. *For any function f constructed by addition, subtraction and multiplication, its floating-point error can be expressed in the form of $\epsilon_f = |f - \hat{f}| \le B_f[(1+\epsilon)^{k_f} - 1]$, where B_f is an upper bound of $|f|$ and k_f is the exponential coefficient. For each operation among addition, subtraction and multiplication, the error bound of the output result o can be calculated by updating B_o and k_o according to its left operand l and right operand r following the rules below:*

(1) If $o = l \pm r$, then $B_o = B_l + B_r$, $k_o = \max(k_l, k_r) + 1$;
(2) If $o = l \times r$, then $B_o = B_l \times B_r$, $k_o = k_l + k_r + 1$.

4.4 Error Analysis for Envelope Construction

We analyze the floating-point error in constructing the envelope $\max_{0\le\alpha<n_1m_1} a_\alpha^{(1)}t + b_\alpha^{(1)}$ while $\min_{0\le\beta<n_2m_2} a_\beta^{(2)}t + b_\beta^{(2)}$ follows analogously. We omit the superscripts for simplicity.

The inexactness in envelope construction arises from two aspects: (1) the coefficients $\{(a_\alpha, b_\alpha)\}$ are obtained as inexact floating-point numbers $\{(\hat{a}_\alpha, \hat{b}_\alpha)\}$; and (2) subsequent floating-point arithmetic operations may lead to wrong decisions when comparing lines.

Let $f(t) = \max_\alpha a_\alpha t + b_\alpha$ be the exact max function computed with exact coefficients, $\hat{f}_1(t) = \max_\alpha \hat{a}_\alpha t + \hat{b}_\alpha$ be the ideal max function using inexact coefficients but exact arithmetic, and $\hat{f}(t) = \hat{\max}_\alpha \hat{a}_\alpha t + \hat{b}_\alpha$ be the function produced by the actual floating-point algorithm. The total error is then bounded as:

$$\begin{aligned}\epsilon_{\text{constr}} &= \max_t |\hat{f} - f| \le \max_t |\hat{f}_1 - f| + \max_t |\hat{f} - \hat{f}_1| \\ &= \epsilon_{\text{coeff}} + \epsilon_{\text{arith}}.\end{aligned} \tag{11}$$

This allows the total error to be analyzed by separating the contribution from coefficient representation and arithmetic computation.

Floating-point coefficients. We first analyze the floating-point error between $f(t)$ and $\hat{f}_1(t)$. For convenience, we here denote the error bounds of $\{a_\alpha\}$ and $\{b_\alpha\}$ by $\epsilon_a = B_a[(1+\epsilon)^{k_a} - 1]$ and $\epsilon_b = B_b[(1+\epsilon)^{k_b} - 1]$ respectively (see Appendix A for details). For $\forall\alpha,\ 0 \le \alpha < n_1m_1$, the error of the line is bounded by

$$\epsilon_{\text{line}} \le \max_t \left|(\hat{a}_\alpha t + \hat{b}_\alpha) - (a_\alpha t + b_\alpha)\right| \le t^{\text{U}}|\hat{a}_\alpha - a_\alpha| + |\hat{b}_\alpha - b_\alpha| \le t^{\text{U}}\epsilon_a + \epsilon_b.$$

Then the difference between $f(t)$ and $\hat{f}_1(t)$ is also bounded by

$$\epsilon_{\text{coeff}} \le \max_{\alpha,t} \left|(\hat{a}_\alpha t + \hat{b}_\alpha) - (a_\alpha t + b_\alpha)\right| \le t^{\text{U}}\epsilon_a + \epsilon_b. \tag{12}$$

Floating-point arithmetic. We then analyze the discrepancy between $\hat{f}(t)$ and $\hat{f}_1(t)$ caused by floating-point algorithmic. As the coefficient inaccuracy has already been considered earlier, we now treat all coefficients $\{(\hat{a}_\alpha, \hat{b}_\alpha)\}$ as exact.

The sorting and duplicate removal do not involve arithmetic calculations and introduce no floating-point errors. Errors are introduced by floating-point subtraction and multiplication in the calculation of Φ and ϕ. By applying Theorem 4.1, the error of the floating-point result Φ^{fp} to the exact result Φ^{ex} when $\gamma > 0$ is bounded by

$$\begin{aligned}\epsilon_\Phi^{\gamma>0} &\le (|\hat{a}_\gamma - \hat{a}_{\gamma-1}||\hat{b}_\alpha - \hat{b}_{\gamma-1}| + |\hat{a}_\alpha - \hat{a}_{\gamma-1}||\hat{b}_\gamma - \hat{b}_{\gamma-1}|)[(1+\epsilon)^4 - 1] \\ &\le (\hat{a}_\alpha - \hat{a}_{\gamma-1})(|\hat{b}_\alpha - \hat{b}_{\gamma-1}| + |\hat{b}_\gamma - \hat{b}_{\gamma-1}|)[(1+\epsilon)^4 - 1],\end{aligned} \tag{13}$$

and when $\gamma = 0$ by

$$\epsilon_\Phi^{\gamma=0} \le (|\hat{a}_\gamma - \hat{a}_\alpha|t^{\text{L}} + |\hat{b}_\gamma - \hat{b}_\alpha|)[(1+\epsilon)^3 - 1]. \tag{14}$$

The error of the floating-point ϕ^{fp} to the exact ϕ^{ex} is bounded by

$$\epsilon_\phi \le (|\hat{a}_\gamma - \hat{a}_\alpha|t^{\text{U}} + |\hat{b}_\gamma - \hat{b}_\alpha|)[(1+\epsilon)^3 - 1]. \tag{15}$$

To ensure robustness, we revise the algorithm in step (2) by modifying the criteria from $\Phi < 0$ and $\phi < 0$ to $\Phi < -\epsilon_\Phi$ and $\phi < -\epsilon_\phi$. This conservative adjustment minimizes the risk of including invalid lines in the convex hull. When the floating-point criterion $\phi^{\text{fp}} < -\epsilon_\phi$ is satisfied, the exact value must satisfies $\phi^{\text{ex}} \le \phi^{\text{fp}} + \epsilon_\phi < 0$, ensuring correctness. Therefore, although this modification may cause the loss of lines, it will prevent the erroneous addition of lines from destroying the convex structure of the envelope, as shown in Fig. 2.

The impact of omitting valid lines is nonetheless bounded. If a line is mistakenly excluded due to the errors in Eq. (14) or Eq. (15), the resulting maximum error between $\hat{f}_1$ and $\hat{f}$ would occur at t^{L} or t^{U}, and is equal to the respective error bound: $\Delta_\Phi^{\gamma=0} = \epsilon_\Phi^{\gamma=0}$ and $\Delta_\phi = \epsilon_\phi$. If a line is wrongly removed from the stack due to the error in Eq. (13), i.e., when $\Phi^{\text{ex}} < 0$ but $\Phi^{\text{fp}} \ge -\epsilon_\Phi^{\gamma>0}$, the exact result would have the lower bound $\Phi^{\text{ex}} > \Phi^{\text{fp}} - \epsilon_\Phi^{\gamma>0} > -2\epsilon_\Phi^{\gamma>0}$. In this case, the maximum error takes place at the intersection point of the α-th line and the $(\gamma-1)$-th line intersect, i.e., at $t_\Phi = -\frac{\hat{b}_\alpha - \hat{b}_{\gamma-1}}{\hat{a}_\alpha - \hat{a}_{\gamma-1}}$, as shown in Fig. 2. So the resulting maximum error is

$$\begin{aligned}\Delta_\Phi &= (\hat{a}_\gamma t_\Phi + \hat{b}_\gamma) - (\hat{a}_{\gamma-1}t_\Phi + \hat{b}_{\gamma-1}) \\ &= \frac{(\hat{a}_\alpha - \hat{a}_{\gamma-1})(\hat{b}_\gamma - \hat{b}_{\gamma-1}) - (\hat{a}_\gamma - \hat{a}_{\gamma-1})(\hat{b}_\alpha - \hat{b}_{\gamma-1})}{\hat{a}_\alpha - \hat{a}_{\gamma-1}} \\ &= -\frac{\Phi^{\text{ex}}}{\hat{a}_\alpha - \hat{a}_{\gamma-1}} \le \frac{2\epsilon_\Phi^{\gamma>0}}{\hat{a}_\alpha - \hat{a}_{\gamma-1}} \le 2(|\hat{b}_\alpha - \hat{b}_{\gamma-1}| + |\hat{b}_\gamma - \hat{b}_{\gamma-1}|)[(1+\epsilon)^4 - 1].\end{aligned}$$

Since each line can be pushed into and popped from the stack at most once, the total number of omission is no more than n_1m_1. Supposing $\Delta_a = \max_{\gamma,\alpha}|\hat{a}_\gamma - \hat{a}_\alpha|$, $\Delta_b = \max_{\gamma,\alpha}|\hat{b}_\gamma - \hat{b}_\alpha|$, then

$$\begin{aligned}\epsilon_{\text{arith}} &\le n_1m_1 \max(\Delta_\Phi^{\gamma=0}, \Delta_\phi, \Delta_\Phi^{\gamma>0}) \\ &\le n_1m_1 \max(4\Delta_b, t^{\text{U}}\Delta_a + \Delta_b)[(1+\epsilon)^4 - 1]\end{aligned} \tag{16}$$

bounds the error between $\hat{f}$ and $\hat{f}_1$ over the entire domain.

Putting them together. To sum up, the approximate max envelope computed by our floating-point algorithm remains close to the exact result. The total deviation is bounded by summing up the error from the two aspects discussed above, as expressed in Eq. (11). To ensure conservativeness under floating error, we shift the computed max envelope upward by ϵ_{constr}, effectively adding ϵ_{constr} to the intercept

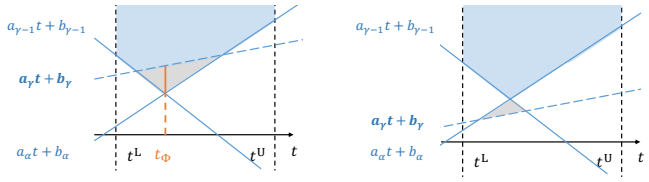

Fig. 2. **Envelope construction errors.** An incorrect decision may either (left) discard a necessary line, enlarging the envelope (denoted by the orange segment at t_Φ) and potentially causing FNs due to reduced collision regions (illustrated in Fig. 1);or (right) retain an extra line, violating convexity. We favor the former for convexity and safeguard FNs through error bounds.

$\hat{b}_\gamma$ of each selected lines. It is worth noting that the error bound and the shift adjustment are themselves computed in floating-point arithmetic. We give the implementation details and prove its safety in Appendix B. Also, a symmetric downward shift is applied to the min envelope. After this correction, we treat the revised coefficients as exact and denote them by $\{c_\alpha\}$ and $\{d_\alpha\}$ for convenience for clarity in the subsequent analysis.

4.5 Error Analysis for Envelope Intersection

Now that the coefficients are treated as exact, the inexactness in envelope intersection arises from subsequent floating-point arithmetic operations. By applying Theorem 4.1, we obtain the error bounds of ψ, $\Psi_{\max}$, $\Psi_{\min}$ and revise the criteria accordingly to $\psi < -\epsilon_\psi$, $\Psi_{\max} < -\epsilon_{\Psi_{\max}}$, and $\Psi_{\min} < -\epsilon_{\Psi_{\min}}$.

Taking Eq. (8a) as an example, $\epsilon_{\Psi_{\max}}$, the error between the floating-point result $\Psi^{\mathrm{fp}}_{\max}$ and the exact result $\Psi^{\mathrm{ex}}_{\max}$ is given by

$$\epsilon_{\Psi_{\max}} = (|c^{(1)}_{\alpha+1} - c^{(2)}_\beta||d^{(1)}_\alpha - d^{(2)}_\beta| + |c^{(1)}_\alpha - c^{(2)}_\beta||d^{(1)}_{\alpha+1} - d^{(2)}_\beta|)[(1+\epsilon)^4 - 1]. \quad (17)$$

We then revise the intersection condition from $\Psi_{\max} < 0$ to $\Psi_{\max} < -\epsilon_{\Psi_{\max}}$, ensuring that when the floating-point predicate returns true, $\Psi^{\mathrm{ex}}_{\max} < \Psi^{\mathrm{fp}}_{\max} + \epsilon_{\Psi_{\max}} < 0$ is always true. Thus, the algorithm would rather miss an intersection (non-collision interval) than wrongly exclude a potential collision. After the modification, a wrong judgment occurs only when $\Psi^{\mathrm{ex}}_{\max} < 0$ but $\Psi^{\mathrm{fp}}_{\max} > -\epsilon_{\Psi_{\max}}$. Fig. 3 illustrates such cases, where errors in Eq. (8a) result in the neglect of the exact intersection between line $c^{(1)}_\alpha t + d^{(1)}_\alpha$ and line $c^{(2)}_\beta t + d^{(2)}_\beta$. If $c^{(2)}_\beta <= c^{(1)}_{\alpha+1}$ (case 1 and 2 in Fig. 3), the error results in extending the potential collision interval, leading to a possible FP—which is acceptable for safety and can thus be neglected. However, if $c^{(2)}_\beta > c^{(1)}_{\alpha+1}$ (case 3 of Fig. 3), the missed intersection may trigger a premature intersection detection in the next iteration, artificially shrinking the solution interval and potentially causing an FN.

A straightforward idea to correct such mistakes would be extending the solution interval $[t^{\mathrm{L}}_{\mathrm{sub}}, t^{\mathrm{U}}_{\mathrm{sub}}]$ with its error bound (Δ_t in case 3 of Fig. 3, highlighted in orange). However, we observe that this time, bounding such arithmetic errors is particularly challenging: when segments are nearly parallel, i.e., $(c^{(2)}_\beta - c^{(1)}_{\alpha+1}) \to 0$, even small numerical errors can be amplified into significant deviations. They require delicate analysis and substantial computational overhead.

To bypass this issue, we adopt a conservative and efficient alternative—computing intersections using slightly pulled-apart envelopes. This mitigates erroneous branching without being affected by error

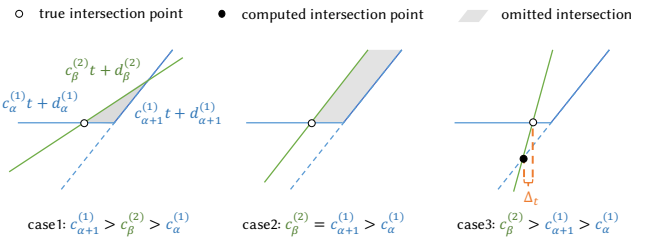

Fig. 3. Three cases when testing $\Psi_{\max}$. When $c^{(2)}_\beta > c^{(1)}_{\alpha+1}$ (left), a wrong judgment may shift the detected point earlier. When $c^{(2)}_\beta \le c^{(1)}_{\alpha+1}$ (middle, right), a wrong judgment may omit an intersection interval.

amplification. Specifically, we introduce an additional offset δ—a pull-up and a pull-down for the max- and min-envelope individually, and compute intersections between the modified envelopes $\{c^{(1)}_\alpha t + \tilde{d}^{(1)}_\alpha\}_\alpha$ and $\{c^{(2)}_\beta t + \tilde{d}^{(2)}_\beta\}_\beta$, where $\tilde{d}^{(1)}_\alpha = d^{(1)}_\alpha + \delta$, $\tilde{d}^{(2)}_\alpha = d^{(2)}_\alpha - \delta$. Supposing $\Delta_d = max_{\alpha,\beta}|d^{(1)}_\alpha - d^{(2)}_\beta|$, we set $\delta = 2\Delta_d[(1+\epsilon)^4 - 1]$. With this setup, the theoretical error is bounded by:

$$\begin{aligned}
\Delta_t &= -\frac{d^{(1)}_\alpha - d^{(2)}_\beta}{c^{(1)}_\alpha - c^{(2)}_\beta} - \left(-\frac{\tilde{d}^{(1)}_{\alpha+1} - \tilde{d}^{(2)}_\beta}{c^{(1)}_{\alpha+1} - c^{(2)}_\beta}\right) \\
&= \frac{-\Psi^{\mathrm{ex}}_{\max} - 2(c^{(2)}_\beta - c^{(1)}_\alpha)\delta}{(c^{(2)}_\beta - c^{(1)}_\alpha)(c^{(2)}_\beta - c^{(1)}_{\alpha+1})} \le \frac{2(\epsilon_{\Psi_{\max}} - (c^{(2)}_\beta - c^{(1)}_\alpha)\delta)}{(c^{(2)}_\beta - c^{(1)}_\alpha)(c^{(2)}_\beta - c^{(1)}_{\alpha+1})} \\
&\le \frac{2}{c^{(2)}_\beta - c^{(1)}_{\alpha+1}}\left((|d^{(1)}_\alpha - d^{(2)}_\beta| + |d^{(1)}_{\alpha+1} - d^{(2)}_\beta|)[(1+\epsilon)^4 - 1] - \delta\right) \\
&\le \frac{2}{c^{(2)}_\beta - c^{(1)}_{\alpha+1}}(2\Delta_d[(1+\epsilon)^4 - 1] - \delta) = 0. \qquad (18)
\end{aligned}$$

A similar derivation applies to Eq. (9a). Therefore even if multiple wrong judgments occur in succession and keep missing intersections, using envelopes pulled apart by $I\delta$ effectively protects against floating-point error, where I is the number of iterations in which intersection is predicated (see Appendix C.3 for a detailed proof).

As for Eq. (7), when a wrong judgment occurs, the true intersection starts at t^1 but the algorithm continues to test successive lines. If $c_0 \le a_0$, the algorithm terminates, missing the intersection but being safe. If $c_0 > a_0$, the computed intersection point may lie before t^1, then clamping it to t^1 still results in a safe interval.

As for Eq. (8b) and (9b), when a wrong judgment occurs, the traversal simply reaches the end of the envelope, so the only consequence is a harmless omission of an intersection.

Finally, the computation of intersection must account for floating-point error as well. We evaluate the associated errors accordingly and implement the computation conservatively, as detailed in Appendix B, enlarging δ and delaying τ^{L} to ensure safety. The final τ^{L} is then treated as a precise safe value for use in the following interval merging and subdivision pipeline.

4.6 A Short Summary

Modifications. Omitting detailed derivations, the safeguarding modifications are:

- For all predicates $\theta < 0$, we compute their cumulative error ϵ_θ and revise them to the conservative form $\theta < -\epsilon_\theta$.

- After computing the envelopes (Step II), we shift the max-envelope upward and the min-envelope downward by ϵ_{constr} respectively.
- When an intersection point is located, we compute it based on the pulled-apart envelopes (additionally offset by δ), conservatively incorporate division-induced errors into the result, and clamp the modified value to be within $[t^{\text{L}}, t^{\text{U}}]$.

Assumptions. This method also adopts the general **assumptions** made by TDIBM:

- Control points move linearly within each time step (Eq. (2)).
- Each surface lies within the convex hull of its control points (convex-hull property of Bézier/B-spline surfaces).

Guarantees. Under the above modifications and assumptions, TDIBM-E is **free of false negatives**: every true collision is detected, even with floating-point implementation.

5 DATASET GENERATION

This section presents a principled methodology for generating collision data with exact ground truth (GT), filling a critical gap where reliable GT benchmarks for CCD of high-order representations are scarce. By reversing the CCD pipeline, we can synthesize a comprehensive range of cases, from randomly distributed "general" collisions to rare and carefully controlled "corner" cases critical for evaluation. This enables a benchmark with both breadth and depth, which are difficult to obtain empirically. We refer to being precise in terms of rational numbers—as done in Wang et al. [2021], all geometric entities and parameters in the construction are represented and manipulated using exact rational arithmetic [Granlund and the GMP development team 2012], eliminating floating-point errors.

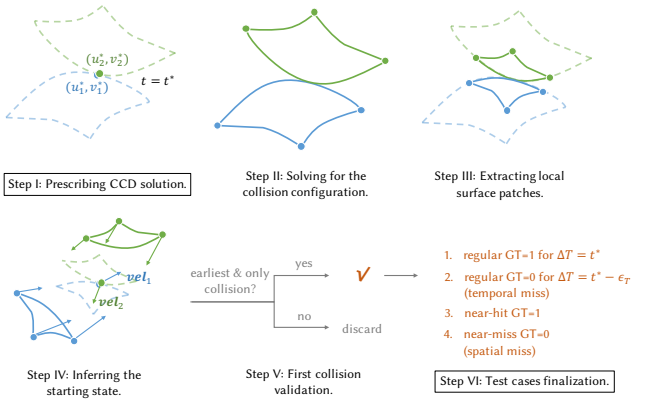

Fig. 4. Pipeline of the dataset generation with ground-truth labels. We use floating-point numbers in the two steps enclosed within black frames and rational numbers in all the others. Specifically, the process starts and ends with floating-point number to construct floating-point test cases, while intermediate computations are performed in rational arithmetic to ensure numerical exactness during construction.

5.1 Inverse Construction for GT Data

Our inverse construction paradigm starts with pre-defined, exact collision outcomes (time, contact points with their parametric coordinates), from which we then derive valid parametric surface configurations—i.e., positions and velocities of the control points of the two surfaces at the start of the time step—that fulfill the collision conditions. To guarantee that the finalized benchmark is losslessly representable in double-precision floating-point format without truncation errors, we execute the entire pipeline using exact rational arithmetic.

We first introduce the overall pipeline, and then detail supplemented treatments for different collision scenarios. Fig. 4 illustrates the following steps in the pipeline:

- **Step I**: Prescribing the CCD solution. We sample a valid CCD solution consisting of the time of impact (ToI) t^* and the parametric coordinates (u_1^*, v_1^*), (u_2^*, v_2^*).
- **Step II:** Constructing the static configurations at the collision moment. We enforce spatial coincidence of S_1 and S_2 must spatially coincide at the prescribed parameters and t^*, by solving a linear constraint system in Eq. (1). Specifically, we leave out a matching number of degrees of freedom (DoFs) unspecified, assign compatible values to the remaining DoFs (see Section 5.4 for details), recovering unspecified DoFs via Gaussian elimination.
- **Step III:** Extracting local surface patches. To avoid unrelated interactions away from the contact region, we extract local surface patches around the collision location, restricting subsequent operations and analysis to their immediate vicinity.
- **Step IV:** Inferring the starting state. To introduce motion, we sample velocity vectors to the collided points on the two surfaces respectively. Since the colliding velocities are interpolations of the control-point velocities, we thus solve for the control-point velocities from the linear interpolation equations similarly to Step II, and then backtrace positions to obtain the initial state.
- **Step V:** First collision validation. To ensure that the prescribed collision at t^* is indeed the *first* one, we further perform a CCD check over the time interval $[0, t^*)$ using TDIBM with rational-arithmetic implementation, and discard cases with earlier collisions. The rational TDIBM will not miss any penetration because TDIBM is theoretically conservative and the rational-number implementation avoids floating-point errors.
- **Step VI:** Test case finalization. Upon successful validation, we save the constructed configuration (control-point position DOFs, velocities and time step $\Delta T = 1$) as the inputs to the CCD query, and equip it with a positive label GT = 1 as the output. To get negative cases (GT= 0), we simply shrink the time step to be $\Delta T = 1 - \epsilon_T$ to avoid collision.

5.2 General Cases

Face-face (FF) collision scenarios represent the most common case in simulation, where collision points (u_1^*, v_1^*) and (u_2^*, v_2^*) lie within the interior domains. To enforce interior collisions, we sample (u_1^*, v_1^*) and (u_2^*, v_2^*) from a restricted sub-domain $(\epsilon_{uv}, 1 - \epsilon_{uv}) \times (\epsilon_{uv}, 1 - \epsilon_{uv})$, where ϵ_{uv} is a small positive rational constant (e.g., 10^{-10}).

In these cases, if the collision between (u_1^*, v_1^*) and (u_2^*, v_2^*) occurs to be the earliest, the normal directions at the two locations must be anti-parallel [Snyder et al. 1993].

We therefore denote the surface normals at the collision points as $\boldsymbol{n}_1(u_1^*, v_1^*, t^*) = -\boldsymbol{n}_2(u_2^*, v_2^*, t^*) = \boldsymbol{n}$, and formulate the corresponding *tangency constraint* as

$$\frac{\partial S_i}{\partial u}(u_i^*, v_i^*, t^*) \cdot \boldsymbol{n} = 0, \quad \frac{\partial S_i}{\partial v}(u_i^*, v_i^*, t^*) \cdot \boldsymbol{n} = 0, \tag{19}$$

where $i = 1, 2$. These equations are incorporated into the linear system in Step II. The normal direction $\boldsymbol{n}$ is sampled beforehand, and all constraints are then solved simultaneously.

To encourage separation before collision, we sample the velocities of the two colliding points within the opposite half-spaces defined by the contact tangent plane.

To accelerate the first collision validation, we apply a quick static penetration check at $t^* = 1$ right after Step II. This can filter out invalid configurations early and significantly improves the overall efficiency (more details are given in Appendix D). These measures ensure efficient, robust generation of diverse yet well-behaved contact scenarios.

5.3 Geometric and Kinematic Corner Cases

Challenging degenerate cases arise when collisions occur on boundaries or exactly at corners of the parametric domain. Based on contact location, we classify such cases into five categories: Edge-Face (EF), Edge-Edge (EE), Vertex-Face (VF), Vertex-Edge (VE), and Vertex-Vertex (VV) collisions. For EF cases, we sample the collision points on an edge of S_1 and in the interior of S_2. We supplement the linear system in Step II with a degenerate version of the tangency constraint that $\boldsymbol{n_2}(u_2^*, v_2^*, t^*)$ is orthogonal to the tangent of the edge on S_1. For the remaining four categories, no additional constraints are imposed due to their geometric specificity. To encourage initial separation, velocities for EF and VF cases are sampled within opposite half-spaces induced by the face tangent plane at contact, while EE cases use the tangent plane spanned by the two incident edges; no velocity restriction is applied to EV or VV cases.

Besides boundary contact cases targeting geometric degeneracies, we additionally consider kinematic degeneracies. For each geometric collision type (FF, EF, EE, VF, VE, VV), we generate corresponding *near-hit* and *near-miss* variants. By constraining the relative velocity at the contact points to lie within the contact tangent plane, we obtain near-hit configurations where surfaces graze past each other while making exact contact. By slightly separating one surface along the contact normal by a small offset, we obtain corresponding near-miss configurations where surfaces narrowly avoid collisions.

5.4 Construction Design for Floating-Point Representation

It is worth noting that, although we perform all the calculations exactly using rational numbers, truncation errors may still occur when converting rational results back to floating-point data inputs in the final step. We next discuss this numerical issue and describe how it is addressed.

A real number is exactly representable in IEEE 754 double-precision format if and only if it can be written as $p/2^k$ for integers p and k with $|p| < 2^{53}$. Such numbers are called *dyadic rationals*. To avoid the aforementioned issue, we require that all quantities throughout the construction pipeline remain dyadic, a property we refer to as the *dyadic property*.

The core challenge of upholding the dyadic property lies in the constraint solving in Step II, which solves for a linear system and may destroy this property. Even if all input elements of the system are dyadic, solving the system through Gaussian elimination produces pivot divisors that are polynomials of the input parameters. These divisors are generally not powers of two, and the resulting solutions will likely lose the dyadic property. In practice, arranging the inputs to achieve cancellation of these complex terms is extremely challenging.

We resolve this by (1) restricting all input parameters to carefully chosen dyadic values, (2) carefully handling the solving process, and (3) adding a round-trip verification as a final safeguard.

Selecting dyadic values. We carefully select all involved quantities to make the constraint-matrix entries remain simple dyadic rational. This increases the probability of finding a valid dyadic solution and reduces the occurring probability of the dyadic property being destroyed. The carefully selected quantities include the below items.

- Control point coordinates and velocities. We either use a standard single-precision floating-point number which is dyadic itself, or draw a integer m from a scaled range and divide by a fixed power of two 2^k, yielding a value of the form $m/2^k$.
- Parametric (UV) coordinates. We restrict UV coordinates to a small predefined set of dyadic rationals for interior contacts, subject to $u + v \leq 1$. Boundary and vertex contacts use analogous dyadic choices or exact domain corners.
- Contact normal construction. We constrain each component to be zero or a signed integer power of two ($\{0, \pm 2^k\}$), with at least one component nonzero.

Solving the system. Rather than solving the full system via Gaussian elimination, we enumerate all candidate 2×2 sub-matrices and solve each via Cramer's rule, accepting the first whose back-substituted residual is exactly zero in rational arithmetic.

Verification. As a final safeguard, every generated configuration undergoes a round-trip test: each rational value r is converted to `double` d and back to rational r'; the case is accepted only if $r = r'$ for all coordinates, velocities, and derived quantities. In our current implementation, the discard rate of this test is 0%, confirming that the above constraints are sufficient to maintain exact representability throughout the pipeline without any post-hoc truncation.

5.5 Implementation and generated dataset

The dataset generation pipeline detailed above yielded a comprehensive benchmark comprising **24246 examples** in total, and all examples were constructed using triangular quadratic Bézier patches and exact rational arithmetic to ensure ground-truth integrity. In table 4, we report the number of examples in each category as well as the average time used to generate one instance per category on a desktop with 12th Gen Intel(R) Core(TM) i7-12700H 2.30 GHz CPU.

Further analysis reveals considerable geometric and kinematic diversity within the benchmark. The generated surfaces exhibit a wide range of local curvatures at the contact points, from nearly planar interactions to highly curved engagements. We illustrate

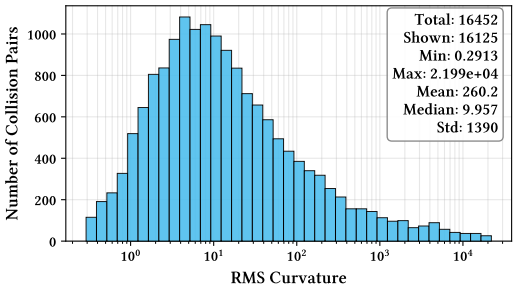

Fig. 5. Distribution of root-mean-square (RMS) curvatures at the contact points of generated collision pairs. The RMS curvature is defined as $\kappa_{\text{RMS}} = \sqrt{(\kappa_1^2 + \kappa_2^2)/2}$, where κ_1, κ_2 are the principal curvatures. The x-axis uses a logarithmic scale and the y-axis shows the number of collision pairs per bin. The top and bottom 1% of values were excluded from the plot.

the curvature distribution in Fig. 5, and show some examples in Fig. 6 and Fig. 7. The variety in both geometry and motion profiles, combined with the guaranteed exactness of the ground truth and the explicit inclusion of difficult corner cases, establishes this benchmark as a unique and powerful tool for in-depth investigation of floating-point robustness in parametric surface CCD.

6 EVALUATION

We evaluate the following four CCD methods on our dataset:

(1) Traditional inclusion-based method (abbreviated as Trad.) [Snyder 1992; Von Herzen et al. 1990], which subdivides the five-dimensional parameter space to locate the solution.
(2) Plain TDIBM, as described in Section 3.
(3) TDIBM with heuristic separation (abbreviated as TDIBM-H), where Chen et al. [2024] proposed a heuristic to slightly pull apart the envelopes to suppress floating-point errors. We here choose pull-apart factors of 10^{-8}, 10^{-12}, and 10^{-16} to evaluate the algorithm's performance.
(4) Our proposed error-bound-enhanced TDIBM (abbreviated as TDIBM-E), which builds upon the plain version and incorporates the improvements summarized in Section 4.6 to rigorously handle floating-point errors.

For methods (1)–(3), we adopt the implementation of Chen et al. [2024]. All the methods use OBB as the inclusion type and adopt the convergence threshold 10^{-4}. We run all the experiments on a 16-core 4.5GHz AMD Ryzen(TM) 9 7950X desktop with 64 GB RAM.

We report the performance statistics on datasets with varying collision scenarios in Table 1, including average runtime, number of false negatives (FN), and number of false positives (FP). Due to the local extraction during data generation, the patches become nearly planar, which is more representative of patches in multi-patch models used in practical applications. However, this also approaches one worst case for the inclusion-based CCD paradigm, where surfaces are nearly coplanar. As a result, compared to Chen et al. [2024], all methods exhibit significantly increased runtime.

Under this setting, the Trad. method incurs unbearable time and memory costs on a large portion of the test cases, and even requires minutes of average runtime on FF cases, which limits this experiment to a reduced subset of the dataset. In contrast, all TDIBM

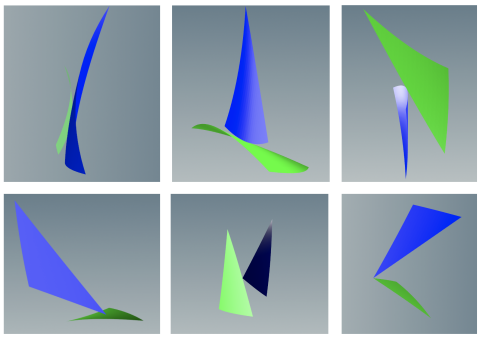

Fig. 6. Examples of the six collision categories, arranged from left to right and top to bottom: FF, EF, EE, VF, VE, VV collision.

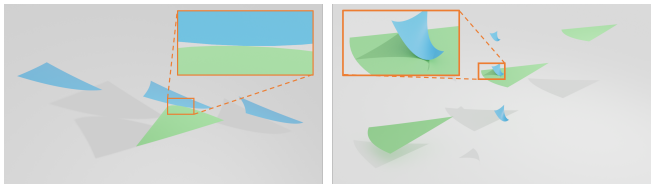

Fig. 7. Examples of near-hit (near-miss) collision cases. The left and right figures illustrate the configurations for edge–edge and face–face collision scenarios, respectively.

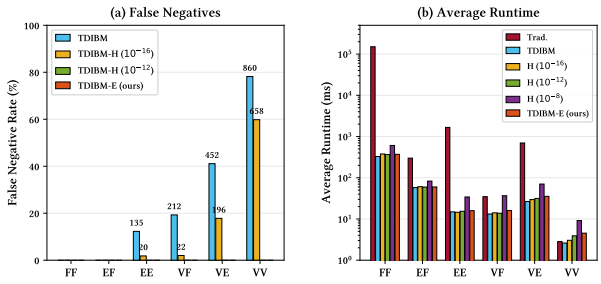

Fig. 8. Visual comparison of CCD methods on our proposed dataset (corresponding to Table 1). (a) False negative rates across six collision scenarios. TDIBM-E and TDIBM-H (10^{-12}) both eliminate all FNs, while TDIBM and TDIBM-H (10^{-16}) exhibit significant FN rates, especially in VV, VE, and VF scenarios. (b) Average runtime on a log scale. TDIBM-E maintains runtime comparable to TDIBM-H (10^{-12}), while the traditional method and TDIBM-H (10^{-8}) are substantially slower.

variants except TDIBM-H with an excessively large pull-apart factor (10^{-8}) consistently kept the runtime below 0.5 seconds across different scenarios. In terms of robustness, both Trad. and TDIBM exhibited FN and FP due to their floating-point implementation. TDIBM-H exposes a clear trade-off between safety and efficiency: small perturbations (10^{-16}) fail to prevent false negatives, whereas large ones (10^{-8}) lead to excessive conservativeness, longer runtimes, and more false positives. The moderate value 10^{-12} suggested in Chen et al. [2024] thus represents only an empirical balance.

In contrast, our error-bound enhanced version achieves both theoretical soundness and practical robustness, eliminating all FNs across the full range of CCD scenarios, with FP rates comparable to or less than TDIBM and TDIBM-H. We provide a visual comparison of the FP and FN statistics across the methods in Fig. 8. The runtime

Table 1. Quantitative comparison of four methods on our proposed dataset. We report average runtime (t) in millisecond, False Positives (FP), and False Negatives (FN).

Scenario	t (ms)						FP						FN					
	Trad.	TDIBM	TDIBM-H			TDIBM-E	Trad.	TDIBM	TDIBM-H			TDIBM-E	Trad.	TDIBM	TDIBM-H			TDIBM-E
			10^{-16}	10^{-12}	10^{-8}				10^{-16}	10^{-12}	10^{-8}				10^{-16}	10^{-12}	10^{-8}	
Face-Face	1.51×10^5	327.73	377.87	366.12	607.99	370.51	13/40	175/6603	175/6603	175/6603	178/6603	175/6603	0/20	0/6603	0/6603	0/6603	0/6603	0/6603
Edge-Face	299.33	57.76	60.78	59.48	82.98	59.85	9/1120	7/1120	7/1120	7/1120	7/1120	7/1120	0/1120	0/1120	0/1120	0/1120	0/1120	0/1120
Edge-Edge	1667.52	14.87	14.53	15.37	34.22	15.89	96/110	121/1100	121/1100	121/1100	121/1100	121/1100	0/110	135/1100	20/1100	0/1100	0/1100	0/1100
Vert-Face	34.75	13.23	14.10	13.80	36.63	16.06	1/1100	0/1100	0/1100	0/1100	0/1100	0/1100	12/1100	212/1100	22/1100	0/1100	0/1100	0/1100
Vert-Edge	698.09	26.55	29.53	31.57	70.36	35.51	458/1100	310/1100	310/1100	312/1100	312/1100	310/1100	73/1100	452/1100	196/1100	0/1100	0/1100	0/1100
Vert-Vert	2.83	2.61	3.04	3.90	9.14	4.53	132/1100	56/1100	56/1100	56/1100	58/1100	56/1100	0/1100	860/1100	658/1100	0/1100	0/1100	0/1100

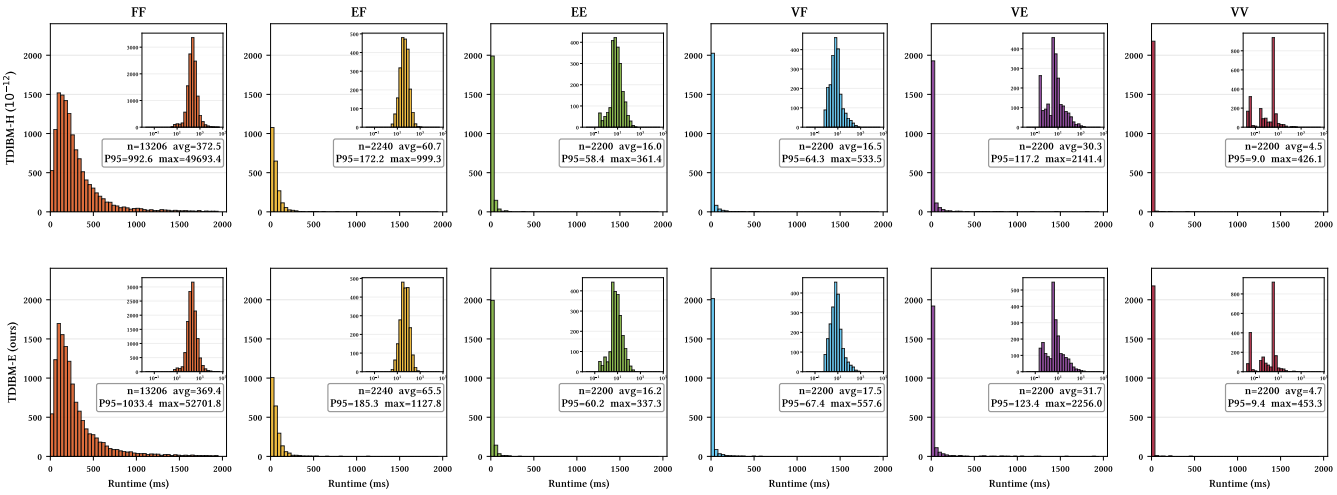

Fig. 9. Runtime distributions of TDIBM-H (10^{-12}) and TDIBM-E (ours) on the complete dataset. Each subplot shows a linear-scale histogram with a log-scale inset (upper-right). The x-axis shows per-query runtime in milliseconds. The two methods exhibit nearly identical distribution shapes, indicating that TDIBM-E's error-bound computation does not cause disproportionate slowdowns even in worst-case scenarios.

overhead remained moderate—no more than 120% over TDIBM and 40% over TDIBM-H. Notably, this overhead is negligible in the more critical cases of FF, EF, EE, and VF, which are more compute-intensive than VE and VV but far more common in simulation and modeling. We also report the per-case runtime distributions of TDIBM-H and TDIBM-E in Fig. 9, demonstrating that the error-bound enhancement generally preserves the runtime behavior of the algorithm, even in worst-case scenarios. These results demonstrate the safety and practicality of our method, offering strong robustness against numerical errors with only minor computational cost. For more details of the experimental results and tests on the CCD dataset of linear mesh, please see Appendix F.

In summary, the proposed benchmark confirms that Trad. is impractical due to excessive computational cost, TDIBM is efficient but prone to false negatives, and TDIBM-H, with a carefully selected parameter, achieves good efficiency and eliminates false negatives but lacks theoretical robustness guarantees. Our method incurs only a slight runtime overhead while providing provable robustness, which is a key advantage for accuracy-critical applications.

7 CONCLUSIONS

We present solutions to the floating-point–induced decision errors arising in CCD for parametric surfaces, including an error-resistant CCD method and a data construction pipeline with known GT for algorithm evaluation under floating-point perturbations. Experimental results demonstrate that the proposed methods are effective and robust, and provide a useful tool for analyzing and improving CCD algorithms in practice. More broadly, we believe that these ideas offer a constructive example of how robustness and evaluation can be approached in geometric algorithms where floating-point errors are unavoidable and exact ground truth is unavailable.

A limitation of the dataset is its focus on isolated point collisions, excluding manifold or area contact scenarios. While TDIBM naturally handles such cases in the same way as single-point contacts, generating dataset containing exact and diverse curve/area contacts would require a fundamentally different algebraic design and more geometric analysis, which is non-trivial. Additionally, the systematic development of more challenging corner cases to rigorously stress-test CCD algorithms remains an open task.Extending the dataset to cover such complex collisions represents an important direction for future work to further advance the robustness and applicability of parametric surface CCD.

Another important future direction would be to extend the method to more complex CAD models for broader applicability.

NURBS.. Standard NURBS can be losslessly converted into tensor-product Bézier patches, which our method natively supports.

Trimmed surfaces. Simple trims (e.g., domain clipping) are directly supported by limiting the parameter query range. For complex trims, since trimming does not affect 3D evaluation, the main difficulty lies in representing and querying the restricted 2D parameter domain, which is a common challenge in intersection detection. TDIBM adopts a divide-and-conquer strategy, which recursively subdivides the parameter space, making it naturally extendable to trimmed surfaces once an appropriate subdivision scheme is designed. We view this as a promising direction for future work.

Multi-patch surfaces. Multi-patch models can be handled patch-by-patch as long as each patch admits such a conversion. For adjacent patches, domain clipping offers an intuitive fix to bypass seams. However, we categorize the problem as self-collision and consider its robust handling another non-trivial future direction.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive comments. This work was supported by the National Natural Science Foundation of China (Project Nos. 62595771 and 62472223). Xingyu Ni also acknowledges the Innovation and Technology Commission of the HKSAR Government under the ITSP-Platform grants (Refs. ITS/335/23FP and ITS/469/24FP).

REFERENCES

Marco Attene. 2020. Indirect Predicates for Geometric Constructions. *Computer-Aided Design* 126 (2020), 102856. https://doi.org/10.1016/j.cad.2020.102856

Tyson Brochu, Essex Edwards, and Robert Bridson. 2012. Efficient geometrically exact continuous collision detection. *ACM Trans. Graph.* 31, 4, Article 96 (jul 2012), 7 pages. https://doi.org/10.1145/2185520.2185592

Christoph Buchenau and Michael Guthe. 2021. Real-Time Curvature-aware Re-Parametrization and Tessellation of Bézier Surfaces. In *Vision, Modeling, and Visualization*, Bjoern Andres, Marcel Campen, and Michael Sedlmair (Eds.). The Eurographics Association. https://doi.org/10.2312/vmv.20211370

Xuwen Chen, Cheng Yu, Xingyu Ni, Mengyu Chu, Bin Wang, and Baoquan Chen. 2024. A Time-Dependent Inclusion-Based Method for Continuous Collision Detection between Parametric Surfaces. *ACM Trans. Graph.* 43, 6, Article 223 (Nov. 2024), 11 pages. https://doi.org/10.1145/3687960

Kenny Erleben. 2018. Methodology for Assessing Mesh-Based Contact Point Methods. *ACM Trans. Graph.* 37, 3, Article 39 (July 2018), 30 pages. https://doi.org/10.1145/3096239

F.R. Feito, C.J. Ogayar, R.J. Segura, and M.L. Rivero. 2013. Fast and accurate evaluation of regularized Boolean operations on triangulated solids. *Computer-Aided Design* 45, 3 (2013), 705–716. https://doi.org/10.1016/j.cad.2012.11.004

Zachary Ferguson, Pranav Jain, Denis Zorin, Teseo Schneider, and Daniele Panozzo. 2023. High-Order Incremental Potential Contact for Elastodynamic Simulation on Curved Meshes. In *ACM SIGGRAPH 2023 Conference Proceedings* (Los Angeles, CA, USA) *(SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA, Article 77, 11 pages. https://doi.org/10.1145/3588432.3591488

A. Galligo and J. P. Pavone. 2006. A sampling algorithm computing self-intersections of parametric surfaces. In *Algebraic Geometry and Geometric Modeling*, Mohamed Elkadi, Bernard Mourrain, and Ragni Piene (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 185–204.

Torbjörn Granlund and the GMP development team. 2012. *GNU MP: The GNU Multiple Precision Arithmetic Library.* https://gmplib.org/ Version 5.0.5.

John Keyser, Tim Culver, Mark Foskey, Shankar Krishnan, and Dinesh Manocha. 2002. ESOLID—A System for Exact Boundary Evaluation. In *Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications* (Saarbrücken, Germany) *(SMA '02)*. Association for Computing Machinery, New York, NY, USA, 23–34. https://doi.org/10.1145/566282.566289

Jia Lu. 2011. Isogeometric contact analysis: Geometric basis and formulation for frictionless contact. *Computer Methods in Applied Mechanics and Engineering* 200, 5 (2011), 726–741.

Jia Lu and Chao Zheng. 2014. Dynamic cloth simulation by isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering* 268 (2014), 475–493.

Zoë Marschner, Paul Zhang, David Palmer, and Justin Solomon. 2021. Sum-of-squares geometry processing. *ACM Trans. Graph.* 40, 6, Article 253 (dec 2021), 13 pages. https://doi.org/10.1145/3478513.3480551

Johnathan Richard Shewchuk. 1996. Robust adaptive floating-point geometric predicates. In *Proceedings of the Twelfth Annual Symposium on Computational Geometry* (Philadelphia, Pennsylvania, USA) *(SCG '96)*. Association for Computing Machinery, New York, NY, USA, 141–150. https://doi.org/10.1145/237218.237337

John M. Snyder. 1992. Interval analysis for computer graphics. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '92)*. Association for Computing Machinery, New York, NY, USA, 121–130. https://doi.org/10.1145/133994.134024

John M. Snyder, Adam R. Woodbury, Kurt Fleischer, Bena Currin, and Alan H. Barr. 1993. Interval methods for multi-point collisions between time-dependent curved surfaces. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (Anaheim, CA) *(SIGGRAPH '93)*. Association for Computing Machinery, New York, NY, USA, 321–334. https://doi.org/10.1145/166117.166158

Stefan Suwelack, Dimitar Lukarski, Vincent Heuveline, Rüdiger Dillmann, and Stefanie Speidel. 2013. Accurate surface embedding for higher order finite elements. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Anaheim, California) *(SCA '13)*. Association for Computing Machinery, New York, NY, USA, 187–192. https://doi.org/10.1145/2485895.2485914

Min Tang, Ruofeng Tong, Zhendong Wang, and Dinesh Manocha. 2014. Fast and exact continuous collision detection with Bernstein sign classification. *ACM Trans. Graph.* 33, 6, Article 186 (nov 2014), 8 pages. https://doi.org/10.1145/2661229.2661237

Brian Von Herzen, Alan H. Barr, and Harold R. Zatz. 1990. Geometric collisions for time-dependent parametric surfaces. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques* (Dallas, TX, USA) *(SIGGRAPH '90)*. Association for Computing Machinery, New York, NY, USA, 39–48. https://doi.org/10.1145/97879.97883

Bolun Wang, Zachary Ferguson, Xin Jiang, Marco Attene, Daniele Panozzo, and Teseo Schneider. 2022. Fast and Exact Root Parity for Continuous Collision Detection. *Computer Graphics Forum (Proceedings of Eurographics)* 41, 2 (2022), 9 pages.

Bolun Wang, Zachary Ferguson, Teseo Schneider, Xin Jiang, Marco Attene, and Daniele Panozzo. 2021. A Large-scale Benchmark and an Inclusion-based Algorithm for Continuous Collision Detection. *ACM Trans. Graph.* 40, 5, Article 188 (sep 2021), 16 pages. https://doi.org/10.1145/3460775

Huamin Wang. 2014. Defending continuous collision detection against errors. *ACM Trans. Graph.* 33, 4, Article 122 (jul 2014), 10 pages. https://doi.org/10.1145/2601097.2601114

Zhendong Wang, Min Tang, Ruofeng Tong, and Dinesh Manocha. 2015. TightCCD: Efficient and Robust Continuous Collision Detection using Tight Error Bounds. *Computer Graphics Forum* 34, 7 (2015), 289–298. https://doi.org/10.1111/cgf.12767 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12767

Wolfram Research, Inc. 2024. *Mathematica, Version 14.0.* Wolfram Research, Inc., Champaign, Illinois. https://www.wolfram.com/mathematica/.

Ruicheng Xiong, Yang Lu, Cong Chen, Jiaming Zhu, Yajun Zeng, and Ligang Liu. 2023. ETER: Elastic Tessellation for Real-Time Pixel-Accurate Rendering of Large-Scale NURBS Models. *ACM Trans. Graph.* 42, 4, Article 133 (jul 2023), 13 pages. https://doi.org/10.1145/3592419

Chee K. Yap. 2004. Robust geometric computation. In *Handbook of Discrete and Computational Geometry, Second Edition*, Jacob E. Goodman and Joseph O'Rourke (Eds.). Chapman and Hall/CRC, 927–952. https://doi.org/10.1201/9781420035315.CH41

Paul Zhang, Zoë Marschner, Justin Solomon, and Rasmus Tamstorf. 2023a. Sum-of-Squares Collision Detection for Curved Shapes and Paths. In *ACM SIGGRAPH 2023 Conference Proceedings* (Los Angeles, CA, USA) *(SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA, Article 76, 11 pages. https://doi.org/10.1145/3588432.3591507

Ran Zhang, Gang Zhao, Wei Wang, and Xiaoxiao Du. 2023b. Large deformation frictional contact formulations for isogeometric Kirchhoff–Love shell. *International Journal of Mechanical Sciences* 249 (2023), 108253. https://doi.org/10.1016/j.ijmecsci.2023.108253

A ACCUMULATED ERRORS IN STEP I

In each iteration, the coefficients of the candidate sub-patches are calculated through (a) subdivision of the parameter interval, (b) interpolation of control points, (c) calculation of OBB axes and (d) projection.

(a) Since each of the sub-patches are checked without omission, inaccurate subdivision does not affect the theoretical accuracy of the algorithm. So we regard $u^{\mathrm{l}}, u^{\mathrm{u}}] \times [v^{\mathrm{l}}, v^{\mathrm{u}}$ as exact numbers.

(b) Given a Bézier path of n-th order, its sub-patch defined by the parameter interval $[u^{\mathrm{l}}, u^{\mathrm{u}}] \times [v^{\mathrm{l}}, v^{\mathrm{u}}] \subseteq [0,1] \times [0,1]$ can be calculated according to de Casteljau's algorithm. Denoting the maximum absolute component value of $\{\boldsymbol{p}_{ij}\}_{ij}$ as P, the control points of the sub-patch $\{\hat{\boldsymbol{q}}_{ij}\}_{ij}$ calculated by floating-point arithmetic differs from its exact value $\{\boldsymbol{q}_{ij}\}_{ij}$ by no more than $\epsilon_{\mathrm{cp}} = B_{\mathrm{cp}}[(1+\epsilon)^{k_{\mathrm{cp}}} - 1]$, where $(B_{\mathrm{cp}}, k_{\mathrm{cp}}) = (P, 6n)$ for quadrilateral patches, and $(B_{\mathrm{cp}}, k_{\mathrm{cp}}) = (2^n P, 5n)$ for triangular patches. The velocities of the sub-patch control points following the same process of interpolating the velocities, and share the same form of error bounds.

(c) As elaborated in Chen et al. [2024], the OBB axes are selected to be aligned with the directions that the sub-patches span at t^{l} moment. Let

$$
\begin{aligned}
\boldsymbol{l}_{u1} &= \boldsymbol{S}_1(u_1^{\mathrm{u}}, v_1^{\mathrm{l}}, t^{\mathrm{l}}) - \boldsymbol{S}_1(u_1^{\mathrm{l}}, v_1^{\mathrm{l}}, t^{\mathrm{l}}),\\
\boldsymbol{l}_{v1} &= \boldsymbol{S}_1(u_1^{\mathrm{l}}, v_1^{\mathrm{u}}, t^{\mathrm{l}}) - \boldsymbol{S}_1(u_1^{\mathrm{l}}, v_1^{\mathrm{l}}, t^{\mathrm{l}}),\\
\boldsymbol{l}_{u2} &= \boldsymbol{S}_2(u_2^{\mathrm{u}}, v_2^{\mathrm{l}}, t^{\mathrm{l}}) - \boldsymbol{S}_2(u_2^{\mathrm{l}}, v_2^{\mathrm{l}}, t^{\mathrm{l}}),\\
\boldsymbol{l}_{v2} &= \boldsymbol{S}_2(u_2^{\mathrm{l}}, v_2^{\mathrm{u}}, t^{\mathrm{l}}) - \boldsymbol{S}_1(u_2^{\mathrm{l}}, v_2^{\mathrm{l}}, t^{\mathrm{l}}).
\end{aligned}
$$

To avoid the modulo operation and since Eq. (3) is linear with the axis length, we calculate the axes as

$$
\begin{aligned}
&\boldsymbol{l}_1 = \boldsymbol{l}_{u1},\ \boldsymbol{l}_2 = \boldsymbol{l}_{u1} \times \boldsymbol{l}_{v1},\ \boldsymbol{l}_3 = \boldsymbol{l}_1 \times \boldsymbol{l}_2,\\
&\boldsymbol{l}_4 = \boldsymbol{l}_{u2},\ \boldsymbol{l}_5 = \boldsymbol{l}_{u2} \times \boldsymbol{l}_{v2},\ \boldsymbol{l}_6 = \boldsymbol{l}_4 \times \boldsymbol{l}_5,\\
&\boldsymbol{l}_{3*i+j+3} = \boldsymbol{l}_i \times \boldsymbol{l}_{3+j}, \quad \text{for} i, j \in 1, 2, 3.
\end{aligned}
$$

Since the above calculation of directions $\{\boldsymbol{l}_{u1}, \boldsymbol{l}_{v1}, \boldsymbol{l}_{u2}, \boldsymbol{l}_{v2}\}$ is prescribed artificially, we regard them as exact numbers and normalize them. Denoting the maximum absolute value of their components as A, the error of the components of the 15 axes are bounded by $\epsilon_{\boldsymbol{l}_r} = B_{\boldsymbol{l}_r}[(1+\epsilon)^{k_{\boldsymbol{l}_r}} - 1]$, where

$$
(B_{\boldsymbol{l}_r},\ k_{\boldsymbol{l}_r}) = \begin{cases} (A, 0), & r = 1, 4\\ (2A^2, 2), & r = 2, 5, 7\\ (4A^3, 4), & r = 3, 6, 8, 10\\ (8A^4, 6), & r = 9, 11, 13\\ (16A^5, 8), & r = 12, 14\\ (32A^6, 10), & r = 15 \end{cases}
$$

(d) For each Without loss of generality, we take Eq. 3a as an example. After projection it is formed as Eq. 4. Then the potential errors for the coefficients $\{a_\alpha^{(1)}\}$, $\{b_\alpha^{(1)}\}$, $\{a_\beta^{(1)}\}$, $\{b_\beta^{(1)}\}$ are bounded by

$$
\begin{aligned}
\epsilon_{a^{(1)}} &= 3 \cdot 2^n B_{\boldsymbol{l}_r} \dot{P}_1[(1+\epsilon)^{6n+k_{\boldsymbol{l}_r}+3} - 1],\\
\epsilon_{b^{(1)}} &= 3 \cdot 2^n B_{\boldsymbol{l}_r} P_1[(1+\epsilon)^{6n+k_{\boldsymbol{l}_r}+3} - 1],\\
\epsilon_{a^{(2)}} &= 3 \cdot 2^n B_{\boldsymbol{l}_r} \dot{P}_2[(1+\epsilon)^{6n+k_{\boldsymbol{l}_r}+3} - 1],\\
\epsilon_{b^{(2)}} &= 3 \cdot 2^n B_{\boldsymbol{l}_r} P_2[(1+\epsilon)^{6n+k_{\boldsymbol{l}_r}+3} - 1].
\end{aligned}
$$

B ROBUST IMPLEMENTATION OF THE ERROR BOUNDS

Although we have derived analytical formulations for error bounds, computing these bounds in floating-point arithmetic introduces additional numerical errors. To ensure safety, we carefully manage their implementation.

For an error bound of the form $\epsilon_x = B_x[(1+\epsilon)^{k_x} - 1]$, we conservatively approximate it as $\epsilon_x' = 2B_x k_x \epsilon$, which overestimates the bound by replacing the exponential term with its linear approximation and applying a factor of 2. This guarantees that the computed floating-point error bound $\hat{\epsilon}_x'$ is safely larger than the true error bound.

When adding the bound to the original value to construct a conservative number (e.g., when pulling the envelopes apart and when calculating safe τ intervals), the result $x + \epsilon_x$ may still suffer from rounding errors. To account for this, we use the `nextafter` function from `<cmath>` to obtain the smallest floating-point number strictly greater than $x + \epsilon_x$. Similarly we use `nextafter` in the opposite direction to safely compute $x - \epsilon_x$ from below. These operations ensure that the conservatively adjusted values remain safe throughout subsequent computations.

C SUPPLEMENTED FORMULATIONS AND EXPLANATION FOR SECTON 4.5

C.1 Error bounds for Eq. (7), (8) and (9)

The errors between the floating-point numbers and exact numbers are bounded by

$$
\epsilon_\psi = (|a_0^{(1)} - a_0^{(2)}|t^{\mathrm{l}} + |b_0^{(1)} - b_0^{(2)}|)[(1+\epsilon)^3 - 1],
$$

$$
\begin{aligned}
\epsilon_{\Psi_{\max}}^{\alpha<\Gamma^{(1)}-1} =& (|a_{\alpha+1}^{(1)} - a_\beta^{(2)}||b_\alpha^{(1)} - b_\beta^{(2)}|\\
&+ |a_\alpha^{(1)} - a_\beta^{(2)}||b_{\alpha+1}^{(1)} - b_\beta^{(2)}|)[(1+\epsilon)^4 - 1],\\
\epsilon_{\Psi_{\max}}^{\alpha=\Gamma^{(1)}-1} =& (|a_\alpha^{(1)} - a_\beta^{(2)}|t^{\mathrm{u}} + |b_\alpha^{(1)} - b_\beta^{(2)}|)[(1+\epsilon)^3 - 1],
\end{aligned}
$$

$$
\begin{aligned}
\epsilon_{\Psi_{\min}}^{\beta<\Gamma^{(2)}-1} =& (|a_\alpha^{(1)} - a_{\beta+1}^{(2)}||b_\alpha^{(1)} - b_\beta^{(2)}|\\
&+ |a_\alpha^{(1)} - a_\beta^{(2)}||b_\alpha^{(1)} - b_{\beta+1}^{(2)}|)[(1+\epsilon)^4 - 1],\\
\epsilon_{\Psi_{\min}}^{\beta=\Gamma^{(2)}-1} =& (|a_\alpha^{(1)} - a_\beta^{(2)}|t^{\mathrm{u}} + |b_\alpha^{(1)} - b_\beta^{(2)}|)[(1+\epsilon)^3 - 1],
\end{aligned}
$$

C.2 Error bounds for Eq. (10)

In Eq. (10), the left endpoint of intersection τ^{L} is reformulated as dividing the left operand $l = \tilde{a}_\alpha^{(1)} - \tilde{a}_\beta^{(2)}$ by the right operands $r = c_\alpha^{(1)} - c_\beta^{(2)}$. The floating-point error introduced by division can

be evaluated as follows:

$$\begin{aligned}\epsilon_\tau^{\mathrm{L}} &= |\hat{l}\hat{\div}\hat{r} - l \div r| \\ &= |(\hat{l}\hat{\div}\hat{r} - \hat{l} \div \hat{r}) + (\hat{l} \div \hat{r} - l \div r)| \\ &\le |(\hat{l}\hat{\div}\hat{r} - \hat{l} \div \hat{r})| + |\frac{1}{\hat{r}}(\hat{l} - l) + \frac{l}{r\hat{r}}(r - \hat{r})| \\ &\le |\frac{\hat{l}}{\hat{r}}|\epsilon + |\frac{1}{\hat{r}}|\epsilon_l + |\frac{l}{r\hat{r}}|\epsilon_r\end{aligned}$$

Since the four coefficients $\tilde{d}_\alpha^{(1)}$, $\tilde{d}_\beta^{(2)}$, $c_\alpha^{(1)}$, $c_\beta^{(2)}$ are all exact numbers, the errors in both operands are introduced solely by the subtraction operations. Therefore, $\epsilon_l = l\epsilon$, $\epsilon_r = r\epsilon$. Then we can simplify the above error bound as

$$\begin{aligned}\epsilon_\tau^{\mathrm{L}} &\le \frac{l(1+\epsilon)}{|\hat{r}|}\epsilon + \frac{1}{|\hat{r}|}l\epsilon + \frac{l}{|\hat{r}|}\epsilon \\ &\le \frac{l}{|\hat{r}|}[(1+\epsilon)^2 - 1 + (1+\epsilon)^1 - 1] \\ &\le \frac{2l}{|\hat{r}|}[(1+\epsilon)^2 - 1].\end{aligned}$$

C.3 Proof of the safety of $I\delta$ in Sec. 4.5

Suppose an intersection is found at the (α^*, β^*)-th segment pair. From $(0, 0)$ to (α^*, β^*), the path consists of a sequence of index pairs, where each transition either increments α (when the $\Psi_{\max} < 0$ criterion fails) or β (when the $\Psi_{\max} < 0$ criterion fails), thereby moving to the next index pair. For every index pair (except the last pair (α^*, β^*)), the situation always corresponds to case 1 in Fig. 3; otherwise, in the other two cases, the algorithm would terminate early, because after the index update, the slope of the min-envelope segment would not exceed that of the max-envelope segment, leading to a non-intersection judgment.

For the J-th and $(J+1)$-th index pairs, we consider the envelopes pulled apart by $2J\delta$ and $2(J+1)\delta$, respectively. Since the offset between them is 2δ, the difference between their corresponding intersection points τ_J and τ_{J+1} satisfy Eq. (18), i.e., $\tau_J - \tau_{J+1} \le 0$. Therefore, for the entire path with a total of $I = a^* + b^*$ steps, the computed intersection would differ from the true intersection point by no more than

$$\tau_0 - \tau_I = (\tau_0 - \tau_1) + \cdots + (\tau_{J-1} - \tau_J) + (\tau_J - \tau_{J+1}) + \cdots + (\tau_{I-1} - \tau_I) \le 0.$$

This indicates that when an intersection is detected in the I-th iteration, shifting each of the envelopes by $I\delta$ ensures the safety—the intersection region (non-collision period) would not be enlarged.

D A QUICK STATIC PENETRATION CHECK

While the tangent planes at the designated collision points are aligned by enforcing the tangency constraint, it is a necessary but not sufficient criterion for the point to be a true time of first contact, because it only affects first-order features. During data generation, we identified counterexamples , where the local geometries at the collision points form saddle shapes, causing the two patches to interpenetrate despite the tangency constraint being satisfied.

Relying solely on rational-version TDIBM CCD filtering to eliminate such cases is inefficient. Therefore, we introduce a preliminary static filtering to exclude them early in the pipeline. The core idea is that we expect the relative offsets of the two patches along the normal to remain consistent along arbitrary tangential direction, i.e., $\boldsymbol{S}_1$ should always stay on one side of $\boldsymbol{S}_2$ around the collision points, as shown in . This requirement, in fact, implicitly involves higher-order geometric information of the surfaces.

Specifically, for arbitrary tangential direction $\boldsymbol{d}$, we first inversely map it back to the parametric domains of the two patches and obtain the corresponding parametric directions $(\Delta u_1, \Delta v_1)$ and $(\Delta u_2, \Delta v_2)$. Taking the first-order approximation of the patches, we set

$$\Delta u_i \frac{\partial \boldsymbol{S}_i}{\partial u}(u_i^*, v_i^*, t^*) + \Delta v_i \frac{\partial \boldsymbol{S}_i}{\partial v}(u_i^*, v_i^*, t^*) = \boldsymbol{d}, \quad i = 1, 2.$$

Since $\boldsymbol{d}$ lies within the plane spanned by $\frac{\partial \boldsymbol{S}_i}{\partial u}$ and $\frac{\partial \boldsymbol{S}_i}{\partial v}$, the least-squares solution to the above linear system gives the accurate answer. Then the normal offset is contributed mainly by the second-order approximation, written as

$$o_i = \boldsymbol{n}_1 \cdot \left[\frac{1}{2}(\Delta u_i)^2 \frac{\partial^2 \boldsymbol{S}_i}{\partial u^2} + \Delta u_i \Delta v_i \frac{\partial^2 \boldsymbol{S}_i}{\partial u \partial v} + \frac{1}{2}(\Delta v_i)^2 \frac{\partial^2 \boldsymbol{S}_i}{\partial v^2}\right], \quad i = 1, 2.$$

We sample multiple tangential directions, compute normal offsets o_1, o_2 of the two patches using the quadratic approximation and then check whether the difference $o_1 - o_2$ maintains a consistent sign across all sampled directions. If the CCD case passes this consistency check, it proceeds to the next step of the pipeline; otherwise, it is discarded.

While this filter is based on a local approximation and may be inaccurate, it does not compromise the correctness of the overall generation pipeline, as a rigorous final validation step is performed at the end. Therefore, it acts as an effective acceleration strategy to reduce the number of problematic configurations early in the process.

E FLOATING-POINT REPRESENTABILITY

This appendix details the additional constraints required to guarantee exact double-precision representability of all generated data.

E.1 Control Point Coordinates and Velocities

To ensure that both geometric configurations and kinematic data satisfy the dyadic property, control point coordinates and their velocity vectors are generated using a unified fixed-point construction. We first sample random integers from a scaled range, such as $[-1000 \times 2^{10}, 1000 \times 2^{10}]$, and then uniformly divide the results by a predefined power of two, such as 2^{15}, which serves as a common denominator. This guarantees that all numerical values defining the geometry and motion are losslessly representable in double-precision floating-point arithmetic from the outset.

E.2 Parametric (UV) Coordinates at the Collision Point

When solving the constraint equations, arbitrary rational UV coordinates at the collision point could act as divisors and introduce non-dyadic values, thereby compromising exact representability. To mitigate this risk, we avoid random sampling and instead **specify** the UV coordinates from a predefined set of dyadic rationals, such as $(\frac{1}{2}, \frac{1}{4})$ or $(\frac{1}{4}, \frac{1}{4})$.

Table 2. Quantitative comparison of four methods on our proposed dataset, including traditional inclusion-based CCD method (Trad.), the base version of TDIBM (TDIBM), and its two enhanced variants—TDIBM-H with envelope-pulling heuristic, and TDIBM-E with error-bound handling (ours). We report average runtime (t) in millisecond, False Positives (FP), and False Negatives (FN).

Scenario	t (ms)						FP						FN					
	Trad.	TDIBM	TDIBM-H			TDIBM-E	Trad.	TDIBM	TDIBM-H			TDIBM-E	Trad.	TDIBM	TDIBM-H			TDIBM-E
			10^{-16}	10^{-12}	10^{-8}				10^{-16}	10^{-12}	10^{-8}				10^{-16}	10^{-12}	10^{-8}	
FF (RP+RN)	2.23×10^5	338.09	390.50	377.61	624.43	382.02	13/20	6/6003	6/6003	6/6003	9/6003	6/6003	0/20	0/6003	0/6003	0/6003	0/6003	0/6003
EF (RP+RN)	293.11	56.06	59.16	57.52	80.02	57.26	2/1020	1/1020	1/1020	1/1020	1/1020	1/1020	0/1020	0/1020	0/1020	0/1020	0/1020	0/1020
EE (RP+RN)	9887.50	12.70	12.39	13.10	31.59	13.27	7/10	50/1000	50/1000	50/1000	50/1000	50/1000	0/10	135/1000	20/1000	0/1000	0/1000	0/1000
VF (RP+RN)	13.91	12.48	13.17	12.95	36.35	15.37	0/1000	0/1000	0/1000	0/1000	0/1000	0/1000	12/1000	212/1000	22/1000	0/1000	0/1000	0/1000
VE (RP+RN)	732.19	27.28	30.12	32.55	72.93	36.57	363/1000	223/1000	223/1000	223/1000	223/1000	223/1000	73/1000	452/1000	196/1000	0/1000	0/1000	0/1000
VV (RP+RN)	2.94	2.68	3.12	4.02	9.50	4.69	65/1000	44/1000	44/1000	44/1000	44/1000	44/1000	0/1000	860/1000	658/1000	0/1000	0/1000	0/1000
FF (NHP+NMN)	8303.91	121.52	125.14	136.12	279.07	140.27	0/20	169/600	169/600	169/600	169/600	169/600	—	0/600	0/600	0/600	0/600	0/600
EF (NHP+NMN)	426.15	92.37	93.76	99.38	143.36	112.65	7/100	6/100	6/100	6/100	6/100	6/100	0/100	0/100	0/100	0/100	0/100	0/100
EE (NHP+NMN)	23.52	58.20	57.37	60.80	87.03	68.31	89/100	71/100	71/100	71/100	71/100	71/100	0/100	0/100	0/100	0/100	0/100	0/100
VF (NHP+NMN)	451.47	28.27	32.69	30.88	42.47	29.78	1/100	0/100	0/100	0/100	0/100	0/100	0/100	0/100	0/100	0/100	0/100	0/100
VE (NHP+NMN)	16.02	11.85	12.23	11.94	19.07	12.14	95/100	87/100	87/100	89/100	89/100	87/100	0/100	0/100	0/100	0/100	0/100	0/100
VV (NHP+NMN)	0.71	1.26	1.37	1.52	1.85	1.29	67/100	12/100	12/100	12/100	14/100	12/100	0/100	0/100	0/100	0/100	0/100	0/100

Table 3. Quantitative comparison of four methods on linear CCD dataset [Erleben 2018; Wang et al. 2021], including traditional inclusion-based CCD method (Trad.), the base version of TDIBM (TDIBM), and its two enhanced variants—TDIBM-H with envelope-pulling heuristic and the pull-apart factor set as 10^{-12}, and TDIBM-E with error-bound handling (ours). We report average runtime (t) in millisecond, False Positives (FP), and False Negatives (FN).

	t				FP				FN			
	Trad.	TDIBM	TDIBM-H	TDIBM-E	Trad.	TDIBM	TDIBM-H	TDIBM-E	Trad.	TDIBM	TDIBM-H	TDIBM-E
Edge-Edge	1.084	2.56×10^{-3}	3.66×10^{-3}	0.21	300/34129	33/34129	117/34129	249/34129	0/34129	338/34129	0/34129	0/34129
Vert-Face	6.13×10^{-3}	5.22×10^{-3}	0.70×10^{-3}	1.72×10^{-2}	94/21209	9/21209	59/21209	107/21209	0/21209	228/21209	0/21209	0/21209

Table 4. The number of examples and the averaged computational time of our dataset. The total time includes the cost of discarded cases that fail the validation checks, remaining practical for constructing a large benchmark.

	FF	EF	EE	VF	VE	VV	total
# RP	6003	1020	1000	1000	1000	1000	11023
# RN	6003	1020	1000	1000	1000	1000	11023
t1	59.03s	26.88s	22.72s	20.83s	20.81s	18.98s	—
# NHP	600	100	100	100	100	100	1100
# NMN	600	100	100	100	100	100	1100
t2	215.23s	131.62s	1249.27s	46.63s	71.21s	24.30s	—

E.3 Contact Normal Construction

As highlighted by the numerical analysis in Section 5, the components of the normal vector $\vec{n}$ are direct contributors to the linear system and have a decisive impact on the numerical properties of the solution. If its components were arbitrary rational numbers, the dyadic property would likely be violated during the solving process. Therefore, we impose a strict constraint on its generation: each component (n_x, n_y, n_z) of the target normal vector is constrained to be a power of two (2^k, where k is an integer). In many cases, we further set it to be an axis-aligned vector (e.g., $(1, 0, 0)$), which maximally simplifies or even decouples the constraint equations, thereby ensuring the numerical safety of the solving process.

E.4 Local Patch Extraction

Maintaining the dyadic property is also crucial during the extraction of local sub-patches. To extract a sub-patch around a collision point, we first construct a relatively large parameter-space sub-triangle centered at the collision point and snap each vertex coordinate to the nearest dyadic rational of the form $m/2^k$. We then verify that the collision point still lies inside the snapped sub-triangle by computing its barycentric coordinates with respect to the vertices. If the snapping causes the collision point to fall outside, we replace it with a smaller sub-triangle whose dyadic vertex offsets are analytically chosen so that the collision point's barycentric coordinates are guaranteed to be positive, ensuring containment by construction. The blossom of the parent Bézier patch is then evaluated at the final dyadic vertices to obtain the sub-patch control points. Since the blossom involves only additions and multiplications of the (dyadic) parent control points and the (dyadic) barycentric coordinates, the sub-patch control points are guaranteed to be dyadic. This preserves numerical integrity throughout the entire generation pipeline.

F DETAILED STATISTICS

F.1 Dataset Generation Statistics

Table 4 reports the number of generated samples and the average generation time. For each geometric configuration, our dataset contains four categories:

(1) Regular Positives (RP): GT = 1, cases where an actual collision occurs;
(2) Regular Negatives (RN): GT = 0, cases without collision due to a shortened query time interval;
(3) Near-Hit Positives (NHP): GT = 1, specially constructed near-hit cases where surfaces graze past each other and just make contact during motion.
(4) Near-Miss Negatives (NMN): GT = 0, specially constructed near-miss cases, where the surfaces pass very close to each other without touching, with small offsets near the intended contact point.

Since category (2) samples are obtained directly from category (1) by making minor modifications and assigning a different GT label, the two categories contain exactly the same number of samples; accordingly, we report a single generation time for both. And, for the same reason, we we report a single generation time for category (3) and (4).

For each geometric configuration, we control the number of near-hit cases and the number of near-miss cases to be one tenth of those of the regular samples. Due to the fact that these configurations are more prone to producing unintended spatial or temporal collisions — which are subsequently discarded — their success rate is lower than that of regular cases, resulting in noticeably longer generation times.

F.2 Breakdown of Detailed Results.

Table 2 provides a more fine-grained breakdown of the performance of different methods on our dataset in Table 1. Given the different construction strategies of the four data categories, we report the statistics for categories (1)(2) and category (3)(4) separately in the upper and lower halves of the table.

For the Trad method, the entry corresponding to near-hit cases under the FN column is left empty, as all test cases exceeded the time limit. When computing the average runtime for other entries involving them, they are therefore excluded from the averaging.

We can draw conclusions consistent with those reported in Section 6 from this table. In addition, the timing results indicate that the TDIBM family exhibits highly stable performance across different contact types, showing comparable behavior on regular cases and their corresponding near-hit or near-miss variants. In contrast, the traditional method exhibits large performance variations: it can be fast on regular cases but significantly slower on the corresponding near-miss variants, and in other configurations shows the opposite trend, indicating a strong sensitivity to subtle changes in collision scenarios and a lack of robustness to small geometric and kinematic perturbations.

F.3 Evaluation on linear-mesh CCD datasets.

Triangle meshes are the most commonly used surface representation. CCD between such meshes is typically decomposed into two degenerate linear cases, namely edge–edge (EE) and vertex–face (VF) interactions. We evaluate different methods on the handcrafted dataset proposed by Wang et al. [2021], which is constructed following the stressful collision scenarios introduced by Erleben [2018], and report the results in Table 3.

The traditional method still exhibits large efficiency variations across different geometric configurations. In contrast, TDIBM and TDIBM-H demonstrate consistently high efficiency. The additional 10^{-12} tolerance introduced in TDIBM-H successfully eliminates the false negatives produced by TDIBM, which is consistent with the findings reported by Chen et al. Our TDIBM-E method, due to its additional computations and increased conservativeness, is slower and generates more false positives than both the original TDIBM and the simple pull-apart variant on these overly simplified geometric configurations. Nevertheless, it effectively resists numerical error perturbations and produces no false negatives.