A Time-Dependent Inclusion-Based Method for Continuous Collision Detection between Parametric Surfaces

XUWEN CHEN, School of Intelligence Science and Technology, Peking University, China
CHENG YU, School of Intelligence Science and Technology, Peking University, China
XINGYU NI, School of Computer Science, Peking University, China
MENGYU CHU, State Key Laboratory of General Artificial Intelligence, Peking University, China
BIN WANG*, State Key Laboratory of General Artificial Intelligence, BIGAI, China
BAOQUAN CHEN*, State Key Laboratory of General Artificial Intelligence, Peking University, China

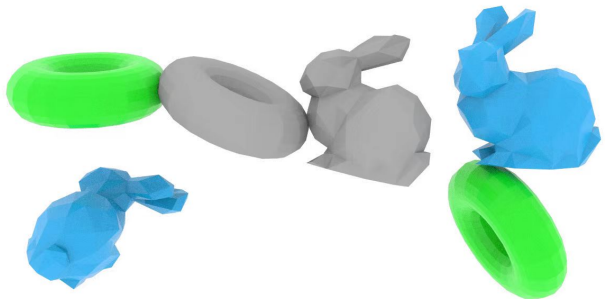


Fig. 1. We solve the CCD problem between a bunny model composed of linear triangles and a torus model composed of rational Bézier patches. The trajectories are obtained by discretizing a rigid motion into time-linear pieces. Our time-dependent inclusion-based method treats all geometric primitives in an efficient and unified way.

Continuous collision detection (CCD) between parametric surfaces is typically formulated as a five-dimensional constrained optimization problem. In the field of CAD and computer graphics, common approaches to solving this problem rely on linearization or sampling strategies. Alternatively, inclusion-based techniques detect collisions by employing 5D inclusion functions, which are typically designed to represent the swept volumes of parametric surfaces over a given time span, and narrowing down the earliest collision moment through subdivision in both spatial and temporal dimensions. However, when high detection accuracy is required, all these approaches significantly increases computational consumption due to the high-dimensional searching space. In this work, we develop a new time-dependent inclusion-based CCD framework that eliminates the need for temporal subdivision and can speedup conventional methods by a factor ranging from 36 to 138. To achieve this, we propose a novel time-dependent inclusion function that provides a continuous representation of a moving surface, along with a corresponding intersection detection algorithm that quickly identifies the time intervals when collisions are likely to occur. We validate our method across various primitive types, demonstrate its efficacy within the simulation pipeline and show that it significantly improves CCD efficiency while maintaining accuracy.

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

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

ACM Reference Format:
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 (December 2024), 11 pages. https://doi.org/10.1145/3687960

1 INTRODUCTION

Continuous collision detection (CCD) has long been a critical procedure in elastodynamic simulation and engineering manufacturing. While CCD techniques for linear triangular meshes have been extensively studied [Bridson et al. 2002; Brochu et al. 2012; Provot 1997; Tang et al. 2014; Teschner et al. 2005; Wang et al. 2021], there has been relatively limited progress in CCD for nonlinear surfaces, such as Bézier and NURBS surfaces. These surfaces are commonly used in computer-aided design (CAD) but present greater mathematical complexity in CCD problems due to their high-order nature.

A 3D parametric surface patch, denoted as $S(u, v) \subset \mathbb{R}^3$, represents a mapping from a two-dimensional parameter space to the three-dimensional world space. When this surface patch is dynamic, it corresponds to a time-dependent mapping $S(u, v, t)$. The CCD problem between two parametric surfaces is typically formulated as a five-variable constrained optimization problem [Snyder et al. 1993; Von Herzen et al. 1990]

$$\min_{t,u_1,v_1,u_2,v_2} \quad t, \tag{1}$$

$$s.t. \quad S_1(u_1, v_1, t) = S_2(u_2, v_2, t), \tag{2}$$

$$0 \le u_1, v_1, u_2, v_2 \le 1, \tag{3}$$

$$0 \le t \le \Delta T. \tag{4}$$

The solution $(t^*, u_1^*, v_1^*, u_2^*, v_2^*)$ to this problem represents to the earliest time of impact (ToI) within the time interval $[0, \Delta T]$, along

*corresponding authors

Authors' addresses: Xuwen Chen, pku_xwchen@163.com, School of Intelligence Science and Technology, Peking University, Beijing, China; Cheng Yu, chengyupku@pku.edu.cn, School of Intelligence Science and Technology, Peking University, Beijing, China; Xingyu Ni, nixy@pku.edu.cn, School of Computer Science, Peking University, Beijing, China; Mengyu Chu, mchu@pku.edu.cn, State Key Laboratory of General Artificial Intelligence, Peking University, China; Bin Wang, binwangbuaa@gmail.com, State Key Laboratory of General Artificial Intelligence, BIGAI, Beijing, China; Baoquan Chen, baoquan@pku.edu.cn, State Key Laboratory of General Artificial Intelligence, Peking University, Beijing, China.

0730-0301/2024/12-ART223 $15.00
https://doi.org/10.1145/3687960

with the parameter coordinates of the collision point on each surface. If no solution exists, it implies that no collision occurs between the two surfaces within $[0, \Delta T]$.

In today's CAD and graphics society, solving CCD problems for high-order parametric surfaces often resorts to either (1) linearization techniques [Ferguson et al. 2023; Xiong et al. 2023], which convert parametric surfaces into linear triangular meshes, or (2) sampling techniques, [Lu 2011; Lu and Zheng 2014] which reduce the problem to CCD between points and a surface. However, since linearized meshes and sampled points only approximate the original surface, the solutions obtained using these techniques are often imprecise, leading to a significant number of false positives (FP) or false negatives (FN). To achieve the level of precision required in high-precision simulations or for large deformations, the resolution of triangulation or sampling strategy must be significantly increased, resulting in substantial inefficiencies. Additionally, a few studies have directly addressed surface-surface CCD problems, including inclusion-based methods [Snyder et al. 1993; Von Herzen et al. 1990] and sum-of-squares programming [Zhang et al. 2023a]. Inclusion-based methods detect the intersection between inclusion functions of surfaces to provided a conservative estimate of collision and recursively subdivides the parameter space to locate the solution. However, since the parameter space of the CCD problem is five-dimensional, this method suffers from a rapid increase in computational time due to the dimensionality of the subdivision. On the other hand, Sum-of-squares programming delicately relaxes the CCD constraints to make them semidefinite and solvable with existing mathematical tools. However, it lacks explicit control over the precision or certification of the solution. Furthermore, as our experiments have shown, it is significantly slower than inclusion-based methods. In summarize, the common drawback of all these methods is their significant time consumption.

In this work, we propose a novel inclusion-based method for efficiently solving CCD problems on parametric surfaces. Unlike traditional approaches that use inclusion functions to bound the entire volume swept by the surface within a time step and then subdivide the five-dimensional space [Snyder et al. 1993; Von Herzen et al. 1990], our method extracts t from the parameter space and perform subdivision only in the four-dimensional parameter space (u_1, v_1, u_2, v_2). To determine the collision time, we introduce a new type of time-dependent inclusion function, along with an algorithm that explicitly computes the intersection period of these inclusion functions. This reduction in problem dimensionality, combined with the rapid narrowing of the candidate temporal domain, significantly decreases computational time and enhances the scalability of our inclusion-based framework, especially as precision requirements increase. Moreover, our method eliminates the need for auxiliary conditions and case-specific discussions, which are often necessary in previous works [Snyder et al. 1993; Zhang et al. 2023a]. Our method is applicable to any surface primitive that satisfies (1) the convex-hull property, where the entire surface lies within the convex hull formed by its control points, and (2) the linear-trajectory assumption, where each control point moves at a constant speed during one time step.

Our contributions are summarized as follows:

(1) We introduce a novel *time-dependent* inclusion function for deforming surfaces and design an algorithm to determine the time intervals during which these functions overlap.
(2) Building on this, we propose a novel paradigm for solving CCD between high-order parametric surfaces, achieving speedups ranging from 36 to 138. The core innovation lies in explicitly calculating and excluding time from the subdivision framework, thereby reducing problem dimensionality and enabling faster convergence in the temporal domain.
(3) We validate our framework across a variety of geometry primitives and collision scenarios, demonstrating significant improvements in both computational time and accuracy.

2 RELATED WORK

CCD for high-order parametric surfaces. The successful application of Isogeometric Analysis (IGA) [Cottrell et al. 2009; Hughes et al. 2005] across various engineering domains has spurred in-depth investigations into collision detection methods [Cardoso and Adetoro 2017; Hughes et al. 1996] for nonlinear surfaces. *Linearization techniques* [Buchenau and Guthe 2021; Suwelack et al. 2013; Xiong et al. 2023] are widely employed in research areas such as CAD, CAE and computer graphics. These techniques transform nonlinear parametric surfaces into linear triangular meshes, enabling the application of well-established CCD algorithms designed for linear meshes[Brochu et al. 2012; Wang et al. 2022, 2021, 2015]. On the other hand, *Sampling techniques* [Galligo and Pavone 2006; Lu 2011; Lu and Zheng 2014; Zhang et al. 2023b] involve sampling points on one surfaces (e.g., fixing u_1, v_1), transforming the problem into detecting collisions between these sampled points and another surface. This approach reduces the optimization problem from five-variable to three, making it more tractable for nonlinear equation solvers. However, both linearization and sampling techniques yield only rough approximate solution. Recently,*Sum-of-squares programming (SOSP)* [Marschner et al. 2021; Zhang et al. 2023a] has emerged as a promising alternative, which directly addresses the nonlinear optimization problem for CCD. SOSP relaxes the collision detection constraints as sum-of-squares (SOS) polynomials, which can then be solved through existing mathematical tools for semidefinite programming(SDP). The exact recovery certificate is often associated with the order of the SOS relaxation; higher orders can potentially provide stronger guarantees of exact recovery, but at the cost of increased computational complexity. To enhance efficiency, the authors have elaborately designed culling strategies[Zhang et al. 2023a] for diverse CCD scenarios. Nevertheless, these strategies still fall short of meeting the real-time requirements of simulations(e.g., 2 seconds for one pair of bicubic triangular patches).

Inclusion-based CCD. The most popular method for solving CCD directly on parametric surfaces [Snyder 1992; Snyder et al. 1993; Von Herzen et al. 1990] is based on inclusion functions and interval analysis, also known as *interval-based methods*. Von Herzen et al. [1990] were the first to combined interval methods with subdivision techniques to address CCD between parametric surfaces. They developed a universal type of bounding volumes for $S(u, v, t)$ based on Lipschitz conditions. Snyder [1992] established the fundamental framework for interval-based methods in computer graphics, and

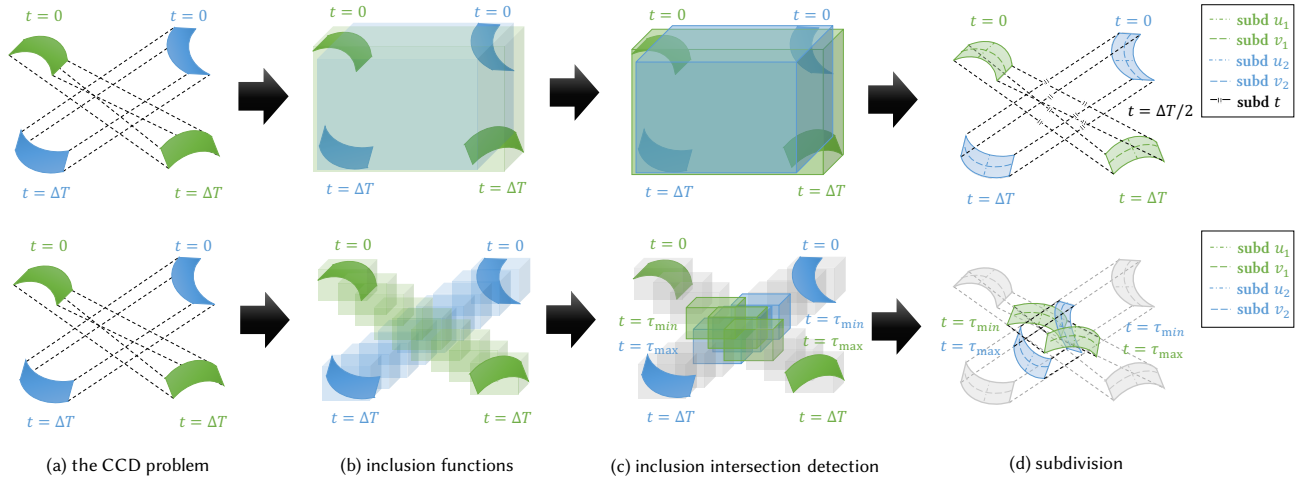


Fig. 2. Pipeline for one iteration of traditional inclusion-based CCD method (top) and our time-dependent inclusion-based CCD method (bottom). Given two surface patches at $t = 0$ and the velocities of their individual control points, the CCD problem, as defined in (a), seeks the earliest time of impact within the time interval $[0, \Delta T]$. Under the linear trajectory assumption, the control points' paths are represented by dashed black lines. For both methods, the main sequential steps are: computing the inclusion functions (b), detecting inclusion intersection (c), and performing subdivision (d). In the traditional method, the inclusion function for each patch is the bounding box of its swept volume within the time interval; while our method uses time-varying bounding box of the patch at each time instance. In (b), these are represented as a single colored box and a series of colored sampling boxes, respectively. As highlighted by the framed colored boxes in (c), the traditional method, by detecting overlaps between the inclusion functions, can only provide Boolean results. In contrast, our method identifies the specific time interval of intersection, discarding the irrelevant intervals (grey boxes). Once potential interpenetration is detected, both methods subdivide their parameter space along the indicated dimensions in (d) for further inspection.

Snyder et al. [1993] extended its application to CCD for parametric surface. However, they discovered that the basic interval method becomes exceedingly slow as precision requirements increase. To mitigate this issue, they supplemented the constraints with tangency conditions and used interval Newton methods to accelerate the process. Despite these improvements, achieving acceptable accuracy (e.g. 10^{-6}) still takes considerable time. Additionally, tangency conditions are only applicable when collisions occur within the surface, necessitating separate treatments for collisions at boundaries and corner vertices. Implementing the interval Newton's method is complex and demands careful handling of degenerate cases, especially when the interval Hessian includes zero. Subsequent application of interval-based methods to CCD problems for linear triangular meshes[Redon et al. 2002; Wang et al. 2021] showed that collisions between linear triangular elements involve only vertex-face (VF) pairs or edge-edge (EE) pairs, both of which require solving only 3 parameters. While this reduces complexity, the interval-based methods have not garnered significant attention due to their relatively slow solving speed compared to other CCD methods for linear meshes. However, in recent years, Wang et al. [2021] revisited the advantages of the interval-based method in avoiding false negatives and carefully designed an interval-based framework for EE tests and VF tests, demonstrating competitive performance with other state-of-the-art methods. Crespel et al. [2024] explored contact detection between high-order fibres via capsule-shaped inclusion functions and highlighted the drawback of linearized elements in contact detection, emphasizing that preserving high-order geometry helps maintain smoothness contact responses. Nevertheless, for nonlinear surfaces, inclusion-based CCD algorithms must contend with a higher subdivision dimension, leading to a significant increase in consumption time as accuracy requirements rise, making them too slow to meet the demands of interactive simulation.

3 BACKGROUND

Inclusion-based methods have utilized interval analysis to address the CCD problem for both linear meshes [Redon et al. 2002; Wang et al. 2021] and parametric surfaces [Snyder et al. 1993; Von Herzen et al. 1990]. We begin with a preliminary introduction to interval arithmetic, drawing on foundational works [Snyder 1992; Snyder et al. 1993], before explaining its application to the CCD problem.

3.1 Interval Arithmetic

A real interval is defined as

$$I = [a, b] = \{x | a \leq x \leq b, x, a, b \in \mathbb{R}\}. \tag{5}$$

More generally, an n-dimensional vector-valued interval is defined as

$$\begin{aligned} \boldsymbol{I} &= [a_1, b_1] \times \cdots \times [a_n, b_n] \\ &= \{\boldsymbol{x} | a_i \leq x_i \leq b_i, x_i, a_i, b_i \in \mathbb{R}, \forall i (1 \leq i \leq n)\}. \end{aligned} \tag{6}$$

The width of such an interval is defined as

$$w(\boldsymbol{I}) = \max_{1 \leq i \leq n} (b_i - a_i). \tag{7}$$

For a vector function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^d$, given an n-dimensional interval $\boldsymbol{I} \subset \mathbb{R}^n$, its *inclusion function* $\square \boldsymbol{f}$ maps the n-dimensional interval

ALGORITHM 1: Inclusion-based CCD solver

Input: two moving surfaces $S_1(u_1, v_1, t)$, $S_2(u_2, v_2, t)$, the initial interval I_0.
Output: the earliest collision time t^*.

Initialize the priority queue $H \leftarrow \emptyset$;
if `IntersectionJudgement`(S_1, S_2, I_0) **then**
  push I_0 into H in asc. order by t^{l};
end
while H *is not empty* **do**
  $I \leftarrow$ pop the beginning element out of H;
  if I *satisfies acceptance criterion* **then**
    return t^{l};
  end
  $\{I^{\mathrm{sub}}\} \leftarrow$Subdivide I into 2^5 sub-intervals;
  foreach I^{sub} **do**
    Calculate the subpatches S_1^{sub}, S_2^{sub} denoted by I^{sub};
    if `IntersectionJudgement`(S_1^{sub}, S_2^{sub}, I^{sub}) **then**
      push I^{sub} into H in asc. order by t^{l};
    end
  end
end
return ∞;

ALGORITHM 2: Time-dependent inclusion-based CCD solver

Input: two moving surfaces $S_1(u_1, v_1, t)$, $S_2(u_2, v_2, t)$, the initial interval $I_0 = \tilde{I}_0 \times [0, \Delta T]$.
Output: the earliest collision time t^*.

Initialize the priority queue $H \leftarrow \emptyset$;
$[\tau_{\min}, \tau_{\max}] \leftarrow$`IntersectionPeriod`($S_1$, S_2, $\tilde{I}_0 \times [0, \Delta T]$);
if $[\tau_{\min}, \tau_{\max}]$ *is not empty* **then**
  push $(\tilde{I}_0 \times [\tau_{\min}, \tau_{\max}])$ into H in asc. order by $\tau_{\min}$;
end
while H *is not empty* **do**
  $I = \tilde{I} \times [t^{\mathrm{l}}, t^{\mathrm{u}}] \leftarrow$ pop the beginning element out of H;
  if I *satisfies acceptance criterion* **then**
    return t^{l};
  end
  $\{\tilde{I}^{\mathrm{sub}}\} \leftarrow$Subdivide $\tilde{I}$ into 2^4 sub-intervals;
  foreach $\tilde{I}_{\mathrm{subd}}$ **do**
    Calculate the subpatches S_1^{sub}, S_2^{sub} denoted by $\tilde{I}^{\mathrm{sub}}$;
    $[\tau_{\min}, \tau_{\max}] \leftarrow$`IntersectionPeriod`($S_1^{\mathrm{sub}}$, S_2^{sub}, $\tilde{I}^{\mathrm{sub}} \times [t^{\mathrm{l}}, t^{\mathrm{u}}]$);
    if $[\tau_{\min}, \tau_{\max}]$ *is not empty* **then**
      push $\tilde{I}^{\mathrm{sub}} \times [\tau_{\min}, \tau_{\max}]$ into H in asc. order by $\tau_{\min}$;
    end
  end
end
return ∞;

to a d-dimensional interval that bounds the range of $\boldsymbol{f}$. Specifically,

$$\boldsymbol{x} \in \boldsymbol{I} \Longrightarrow \boldsymbol{f}(\boldsymbol{x}) \in \square \boldsymbol{f}(\boldsymbol{I}). \tag{8}$$

If, as $\boldsymbol{I}$ converges to a single point $\boldsymbol{p}$, the inclusion function $\square \boldsymbol{f}(\boldsymbol{I})$ also converges to $\boldsymbol{f}(\boldsymbol{p})$, we say that $\square \boldsymbol{f}$ is *convergent*.

3.2 Interval Arithmetic in CCD Problems

As discussed above, CCD for parametric surfaces is mathematically formulated as a minimization problem defined over a 5-dimensional interval

$$\begin{aligned} \boldsymbol{x} \in \boldsymbol{I} &= I_{u_1} \times I_{v_1} \times I_{u_2} \times I_{v_2} \times I_t \\ &= [u_1^{\mathrm{l}}, u_1^{\mathrm{u}}] \times [v_1^{\mathrm{l}}, v_1^{\mathrm{u}}] \times [u_2^{\mathrm{l}}, u_2^{\mathrm{u}}] \times [v_2^{\mathrm{l}}, v_2^{\mathrm{u}}] \times [t^{\mathrm{l}}, t^{\mathrm{u}}], \end{aligned} \tag{9}$$

with the objective function and constraints defined as

$$f(\boldsymbol{x}) = t, \tag{10}$$

$$\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{S}_1(u_1, v_1, t) - \boldsymbol{S}_2(u_2, v_2, t) = \boldsymbol{0}. \tag{11}$$

Using interval arithmetic, the exact constraint in Eq. 11 is relaxed to an intersection condition between the inclusion functions of the two surfaces, which is,

$$\square \boldsymbol{S}_1(\boldsymbol{I}) \cap \square \boldsymbol{S}_2(\boldsymbol{I}) \neq \emptyset. \tag{12}$$

When addressing the CCD problem, an inclusion-based method focuses on reducing the uncertainty in the position and timing of potential collisions from the entire definition domain to a sufficiently small interval. The algorithm pipeline is summarized in Alg. 1. Starting from the initial interval $\boldsymbol{I}_0 = [0, 1] \times [0, 1] \times [0, 1] \times [0, 1] \times [0, \Delta T]$, the algorithm checks whether the inclusions corresponding to the two surfaces intersect. If they do not intersect, a collision is not possible, and the interval can be discarded. If an intersection is detected, indicating a potential collision, the algorithm subdivides the intervals into smaller sub-intervals to refine the estimation of the collision point. This refinement process involves dividing the intervals along each dimension (e.g., time, spatial coordinates) and recalculating the inclusions for these smaller intervals. The resulting sub-intervals are then added to a priority queue, which is sorted by the lower bound of $\square f = [t^{\mathrm{l}}, t^{\mathrm{u}}]$ in ascending order. This process continues recursively until either all intervals in the queue are discarded, indicating that $\boldsymbol{I}_0$ contains no feasible solution, or the subdivision meets the acceptance criterion, thereby identifying a solution to the problem. Since Eq. (12) provides a necessary condition for the intersection of the two surfaces, the inclusion-based method avoids false negatives (excluding the effects of floating-point errors), ensuring there is no interpenetration. The priority queue's organization ensures that the resulting solution is a global minimizer.

In practical implementation, the inclusion function for a surface can be selected from various options, including, but not limited to, bounding boxes and bounding spheres, as long as the chosen function is convergent. As illustrated in the first row of Fig. 2, we employ the bounding boxes of the sweeping volumes of $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ as their respective inclusion functions, $\square \boldsymbol{S}_1$ and $\square \boldsymbol{S}_2$. *The acceptance criterion* [Snyder et al. 1993; Wang et al. 2021] is typically defined as $w(\boldsymbol{I}) < \delta$, where δ is a specified precision tolerance. An alternative approach, proposed by Snyder et al. [1993] involves bounding the surface inclusions such that $\max\{w(\square \boldsymbol{S}_1(\boldsymbol{I})), w(\square \boldsymbol{S}_2(\boldsymbol{I}))\} < \delta$. Wang et al. [2021] introduced a new criterion that estimates the interval width in the co-domain. For simplicity, we adopt the criterion $w(\boldsymbol{I}) < \delta$ in our implementation.

4 TIME-DEPENDENT INCLUSION-BASED CCD

While interval arithmetic in CCD ensures robust and accurate collision detection, its inherent conservativeness and the high subdivision dimension can lead to low computational efficiency. We propose a novel paradigm based on time-dependent inclusion function. Instead of determining $\square f = [t^{\mathrm{l}}, t^{\mathrm{u}}]$ by subdividing the interval along the t-dimension, we reformulate the constraints as functions of t, thereby computing $\square f$ from the constraints.

4.1 Time-Dependent Framework

Specifically, we re-define the CCD problem over a 4-dimensional intervals, denoted by:

$$\begin{aligned}\tilde{\boldsymbol{x}} \in \tilde{\boldsymbol{I}} &= I_{u_1} \times I_{v_1} \times I_{u_2} \times I_{v_2} \\ &= [u_1^{\mathrm{l}}, u_1^{\mathrm{u}}] \times [v_1^{\mathrm{l}}, v_1^{\mathrm{u}}] \times [u_2^{\mathrm{l}}, u_2^{\mathrm{u}}] \times [v_2^{\mathrm{l}}, v_2^{\mathrm{u}}],\end{aligned} \tag{13}$$

with the initial interval $\tilde{\boldsymbol{I}}_0 = [0,1] \times [0,1] \times [0,1] \times [0,1]$. The tilde notation indicates modified variables/intervals/inclusions within the 4D parameter space. Each surface inclusion is then reformulated as a continuous, time-dependent function $\tilde{\square}\boldsymbol{S}(\tilde{\boldsymbol{I}}, t)$, as shown in the second row of Fig. 2. To verify the existence of a feasible domain within $\tilde{\boldsymbol{I}}$, we check whether $\tilde{\square}\boldsymbol{S}_1(\tilde{\boldsymbol{I}}, t)$ intersects $\tilde{\square}\boldsymbol{S}_2(\tilde{\boldsymbol{I}}, t)$ and solve for the feasible interval of $\tilde{\square}t$, defined as:

$$\tilde{\square}t = \{t | \tilde{\square}\boldsymbol{S}_1(\tilde{\boldsymbol{I}}, t) \cap \tilde{\square}\boldsymbol{S}_2(\tilde{\boldsymbol{I}}, t) \neq \emptyset\}. \tag{14}$$

As demonstrated in Fig. 2, our method differs from the traditional method in two significant ways: (1) Our continuous time-dependent inclusion function tightly bounds the patch at each moment, as oppsed to using a bounding box for the sweeping volume over the entire time interval. (2) After determining the intersection of the inclusion functions, the traditional method subdivides both spatial and temporal dimensions, whereas our method directly identifies the sub-time interval where the collision occurs, necessitating subdivision only in the spatial dimensions. Although the per-iteration computation of our method is higher, the total number of iterations is significantly reduced, thereby enhancing the efficiency of CCD. The pipeline of our CCD algorithm is summarized in Alg. 2, with differences from the traditional method highlighted in blue.

A critical step in our CCD method is detecting when the two time-dependent inclusion functions intersect, i.e., solving for the time inclusion in Eq. (14). To address this, assuming the convex-hull property and linear-trajectory motion, we design a new intersection detection algorithm tailored to our proposed time-dependent inclusion functions, as detailed in the following subsections.

4.2 Time-Dependent Inclusion Functions

The tensor-product Bézier patch is a widely used geometric primitive in CAD. In the following, we use Bézier patches to illustrate the construction of our proposed time-dependent inclusion functions and the necessary conditions for their intersection. This approach also applies to other geometric primitives that satisfy the convex-hull property and linear-trajectory motion.

A Bezier patch of order $n \times m$ can be expressed as:

$$\boldsymbol{S}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_i^n(u) B_j^m(v) \boldsymbol{p}_{ij}, \tag{15}$$

where $u, v \in [0,1]$ are the parametric coordinates, $B_i^n(u)$, $B_j^m(v)$ are the Bernstein basis functions, and $\boldsymbol{p}_{ij}$ are the control points. Assuming each control point moves at a constant speed during the time step $t \in [0, \Delta T]$, the corresponding moving surface is represented as:

$$\boldsymbol{S}(u,v,t) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_i^n(u) B_j^m(v) (\boldsymbol{p}_{ij} + t\dot{\boldsymbol{p}}_{ij}), \tag{16}$$

where $\dot{\boldsymbol{p}}_{ij}$ are the velocities of the control points.

According to the convex-hull property of Bézier patches, the axis-aligned bounding box (AABB) of each patch can be determined by projecting the control points onto each coordinate axis to find the maximum and minimum extents in each direction. Therefore, the AABB-based inclusion function at time t for the moving patch defined in Eq.(16) is given by:

$$\tilde{\square}\boldsymbol{S}(\tilde{\boldsymbol{I}}, t) = \{\boldsymbol{x} | \min_{ij}(\boldsymbol{p}_{ij} + t\dot{\boldsymbol{p}}_{ij}) \leq \boldsymbol{x} \leq \max_{ij}(\boldsymbol{p}_{ij} + t\dot{\boldsymbol{p}}_{ij})\}, \tag{17}$$

where the inequality applies to each component of the vectors. The necessary and sufficient condition for the collision of two AABB-based inclusion functions at time t is that their extents overlap along all the axes. This condition is expressed as:

$$\min_{ij}[(\boldsymbol{p}_{ij}^{(2)} + t\dot{\boldsymbol{p}}_{ij}^{(2)}) \cdot \boldsymbol{e}_k] \leq \max_{ij}[(\boldsymbol{p}_{ij}^{(1)} + t\dot{\boldsymbol{p}}_{ij}^{(1)}) \cdot \boldsymbol{e}_k], \tag{18a}$$

AND

$$\min_{ij}[(\boldsymbol{p}_{ij}^{(1)} + t\dot{\boldsymbol{p}}_{ij}^{(1)}) \cdot \boldsymbol{e}_k] \leq \max_{ij}[(\boldsymbol{p}_{ij}^{(2)} + t\dot{\boldsymbol{p}}_{ij}^{(2)}) \cdot \boldsymbol{e}_k], \tag{18b}$$

where the superscripts (1) and (2) indicate which patch the control points belong to, and $\boldsymbol{e}_k \in \{\boldsymbol{e}_0, \boldsymbol{e}_1, \boldsymbol{e}_2\}$ denotes the three standard basis vectors in the world space. Within each min/max operator, $(\boldsymbol{p}_{ij} + t\dot{\boldsymbol{p}}_{ij}) \cdot \boldsymbol{e}_k$ describes the trajectory projection of the ij-th control point onto the axis $\boldsymbol{e}_k$, which is a linear function with respect to t.

Oriented bounding boxes (OBBs) typically provide tighter bounds for patches than AABBs. For OBB-based inclusion functions, the intersection condition follows the same mathematical form as Eq. (18), with the only difference being the selection of $\boldsymbol{e}_k$. Let the axes of the OBBs for the two Bezier patches be denoted as $\{\hat{\boldsymbol{l}}_1^{(1)}, \hat{\boldsymbol{l}}_2^{(1)}, \hat{\boldsymbol{l}}_3^{(1)}\}$ and $\{\hat{\boldsymbol{l}}_1^{(2)}, \hat{\boldsymbol{l}}_2^{(2)}, \hat{\boldsymbol{l}}_3^{(2)}\}$ respectively. The vectors $\boldsymbol{e}_k$s are the 15 axes $\{\hat{\boldsymbol{l}}_i^{(1)}, \hat{\boldsymbol{l}}_j^{(2)}, \ \hat{\boldsymbol{l}}_i^{(1)} \times \hat{\boldsymbol{l}}_j^{(2)} | \ i, j \in \{1,2,3\}\}$, defined by the separating axis theorem [Gottschalk 1996]. During implementation, instead of computing $\{\hat{\boldsymbol{l}}_1, \hat{\boldsymbol{l}}_2, \hat{\boldsymbol{l}}_3\}$ using the computationally expensive singular value decomposition (SVD), we directly assign them to align with the mapped directions of the parameter coordinates [Efremov et al. 2005] as follows:

$$\boldsymbol{l}(u) = [(S(1,0) - S(0,0)) + (S(1,1) - S(0,1))]/2, \tag{19}$$

$$\boldsymbol{l}(v) = [(S(0,1) - S(0,0)) + (S(1,1) - S(1,0))]/2. \tag{20}$$

Then, the OBB axes are:

$$\hat{\boldsymbol{l}}_1 = \boldsymbol{l}(u), \ \hat{\boldsymbol{l}}_2 = \boldsymbol{l}(u) \times \boldsymbol{l}(v), \ \hat{\boldsymbol{l}}_3 = \hat{\boldsymbol{l}}_1 \times \hat{\boldsymbol{l}}_2. \tag{21}$$

For a moving patch, we use its intial shape at each time step to compute its OBB axes and keep them fixed within that time step.

When excluding floating-point errors, the condition in Eq. (18) is both (1) conservative, because the intersection between inclusion functions is a sufficient condition for a collision between patches,

ALGORITHM 3: Intersection Period

Input: two moving surfaces $S_1(u_1, v_1, t)$, $S_2(u_2, v_2, t)$, the candidate interval $\boldsymbol{I} = \tilde{\boldsymbol{I}} \times [t^{\mathrm{l}}, t^{\mathrm{u}}]$.

Output: time interval $[\tau_{\min}, \tau_{\max}]$ when the two inclusions intersect.

Discretely represent $S_1(u_1, v_1, t)$ as $\{\boldsymbol{p}_{ij}^{(1)}\}_{ij}$ and $\{\dot{\boldsymbol{p}}_{ij}^{(1)}\}_{ij}$;
Discretely represent $S_2(u_2, v_2, t)$ as $\{\boldsymbol{p}_{ij}^{(2)}\}_{ij}$ and $\{\dot{\boldsymbol{p}}_{ij}^{(2)}\}_{ij}$;
Select bounding-box axes $\{\boldsymbol{e}_k\}$;
Initialize the set of non-intersection time intervals $\Gamma \leftarrow \emptyset$;
foreach $\boldsymbol{e}_k$ **do**
    Calculate the line set $\mathcal{L}_1 \leftarrow \{(\boldsymbol{p}_{ij}^{(1)} + t\dot{\boldsymbol{p}}_{ij}^{(1)}) \cdot \boldsymbol{e}_k\}_{ij}$;
    Calculate the line set $\mathcal{L}_2 \leftarrow \{(\boldsymbol{p}_{ij}^{(2)} + t\dot{\boldsymbol{p}}_{ij}^{(2)}) \cdot \boldsymbol{e}_k\}_{ij}$;
    Calculate the upper boundary $\mathcal{B}_1 \leftarrow \mathtt{Max}(\mathcal{L}_1, [t^{\mathrm{l}}, t^{\mathrm{u}}])$;
    Calculate the lower boundary $\mathcal{B}_2 \leftarrow \mathtt{Min}(\mathcal{L}_2, [t^{\mathrm{l}}, t^{\mathrm{u}}])$;
    $\Gamma \leftarrow \Gamma \bigcup \mathtt{BoundaryIntersect}(\mathcal{B}_1, \mathcal{B}_2, [t^{\mathrm{l}}, t^{\mathrm{u}}])$;
    Calculate the lower boundary $\mathcal{B}_1 \leftarrow \mathtt{Min}(\mathcal{L}_1, [t^{\mathrm{l}}, t^{\mathrm{u}}])$;
    Calculate the upper boundary $\mathcal{B}_2 \leftarrow \mathtt{Max}(\mathcal{L}_2, [t^{\mathrm{l}}, t^{\mathrm{u}}])$;
    $\Gamma \leftarrow \Gamma \bigcup \mathtt{BoundaryIntersect}(\mathcal{B}_2, \mathcal{B}_1, [t^{\mathrm{l}}, t^{\mathrm{u}}])$;
end
$\tau_{\min} \leftarrow \mathtt{LB}([t^{\mathrm{l}}, t^{\mathrm{u}}]/\Gamma)$;
$\tau_{\max} \leftarrow \mathtt{UB}([t^{\mathrm{l}}, t^{\mathrm{u}}]/\Gamma)$;
return $[\tau_{\min}, \tau_{\max}]$;

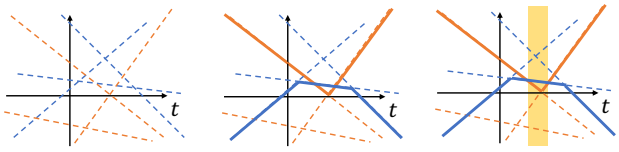


Fig. 3. The process for solving a single inequality in Eq. 18. The linear functions within the min function are represented by blue dashed lines, while those within the max function are shown as orange dashed lines. We first calculate the minimum function (bold blue contour) and the maximum function (bold orange contour) respectively, and then identify the t interval where they intersect and the inequality fails (yellow range). The solution to the inequality is the complement of this yellow range.

and (2) convergent, because the bounding box computed via control points shrinks to the same point as the subdivision converges.

4.3 Inclusion Intersection Detection

The task of inclusion intersection detection involves identifying a compact time interval $[\tau_{\min}, \tau_{\max}]$ within a given time range $[t^{\mathrm{l}}, t^{\mathrm{u}}]$ that encompasses all moments when the inequalities in Eq. 18 are simultaneously satisfied across all directions $\boldsymbol{e}_k$.

For a specific direction $\boldsymbol{e}_k$, the expression $\{(\boldsymbol{p}_{ij}^{(\alpha)} + t\dot{\boldsymbol{p}}_{ij}^{(\alpha)}) \cdot \boldsymbol{e}_k\}$ forms a collection of linear functions with respect to t for all control points $\boldsymbol{p}_{ij}$ belonging to object α, $\alpha \in \{1, 2\}$, as illustrated by the dotted lines of identical color in Fig. 3 (left). The min and max operations on these dotted lines determine the lower and upper boundaries, respectively, as depicted by the solid contours of the same color in Fig. 3 (middle).

Therefore, analyzing when the inequality holds is equivalent to checking when the lower boundary of one patch lies below the upper boundary of the other patch. Apart from the scenario where the lower boundary consistently remains below the upper boundary throughout the entire time range, there may be instances where the lower boundary intersects with the upper boundary, as illustrated in Fig. 3 (right). Given that the min function is concave and the max function is convex, the "lying-below" can only be violated within a single connected interval. The exact moment when the lower boundary crosses the upper boundary can be easily determined through 2D line segment intersection detection.

In parctical implementation, by traversing all separation axes, we obtain a collection of time intervals where at least one inequality in Eq. 18 does not hold. The remaining subintervals within the time range $[t^{\mathrm{l}}, t^{\mathrm{u}}]$ are the moments when Eq. 18 holds, indicating potentially intersections between the two inclusions. To maintain simplicity and conservativeness, we use a single interval that tightly encompasses all these subintervals as the final solution, denoted as $[\tau_{\min}, \tau_{\max}]$. This interval represents the period during which the two patches under observation may collide during their movement.

The overall algorithm is shown in Alg. 3. Suppose the two patches involved in the CCD problem are of order $n_1 \times m_1$ and $n_2 \times m_2$, with the number of control points being $N_1 = (n_1 + 1) \times (m_1 + 1)$ and $N_2 = (n_2 + 1) \times (m_2 + 1)$ respectively. The time complexity of the algorithm is $O(N_1 \log N_1 + N_2 \log N_2)$. Detailed implementation and analysis are provided in the supplementary material.

To enhance the algorithm's robustness against floating errors in practical implementation, we adopt the following strategies: (1) we extend the candidate time interval $[t^{\mathrm{l}}, t^{\mathrm{u}}]$ by a small margin, such as 10^{-6}, to ensure that the endpoints of the time interval are correctly processed; and (2) we increase the intercept of each line within the max function by one millionth to slightly raise the max-function contour, and similarly adjust the min-function contour downward. While these techniques do not provide a theoretical guarantee of safety, they have proven effective in eliminating all false negatives in practice, as demonstrated in the subsequent section.

5 RESULTS

To validate the efficiency, robustness and generality of our method, we conducted a comprehensive series of validation tests and comparison experiments. All experiments were performed in a single-thread mode on a 16-core 4.5GHz AMD Ryzen(TM) 9 7950X desktop with 64 GB RAM.

5.1 Validation

We compare the performance of our method against two prevailing CCD methods:

(1) The traditional inclusion-based method (abbreviated as Trad.) described in Alg. 1. For the sake of fairness, we apply the same subdivision strategy (bisecting all the parameter dimensions in each iteration) and acceptance criterion ($w(\boldsymbol{I}) < \delta$) for both Trad. and our method. Our implementation of Trad. excludes the tangency conditions and the Interval Newton's method proposed by Snyder et al. [1993]. Comparisons are performed using both AABB and OBB inclusion functions, under varying precision tolerance $\delta = 10^{-4}, 10^{-5}, 10^{-6}$.

Table 1. Average time cost [ms] of 1000 random cases using traditional inclusion-based method (Trad.), SOSP and our method under different precision tolerances. Experiments are conducted on triangular patches (T.) and quadrilateral patches (Q.) of order 1, 2, 3, respectively. The number of collision pairs (CP) for each primitive setting is also reported.

			Trad. (AABB)			Ours (AABB)			Trad. (OBB)			Ours (OBB)			SOSP
	d	CP	10^{-4}	10^{-5}	10^{-6}	10^{-4}	10^{-5}	10^{-6}	10^{-4}	10^{-5}	10^{-6}	10^{-4}	10^{-5}	10^{-6}	–
T.	1	468	8.15	13.48	19.57	18.00	32.70	49.70	3.45	4.93	6.85	4.26	5.51	6.68	175.02
	2	776	159.07	1145.41	7748.46	215.16	1489.60	11934.78	61.20	374.81	3264.07	22.40	42.67	90.87	251.00
	3	932	523.10	3138.51	26564.17	375.81	2610.75	19525.31	203.21	1226.84	9974.95	35.98	66.73	131.30	8755.73
Q.	1	670	21.38	99.53	836.18	21.93	83.16	443.23	12.70	44.04	619.46	5.00	9.72	22.13	309.35
	2	933	333.06	2233.37	16196.34	205.02	1381.54	10934.02	133.04	891.13	6858.91	19.30	32.50	62.12	784.52
	3	988	730.90	4875.23	35332.08	337.91	2336.47	17901.88	205.46	1363.95	10962.40	27.17	44.71	79.40	56056.00

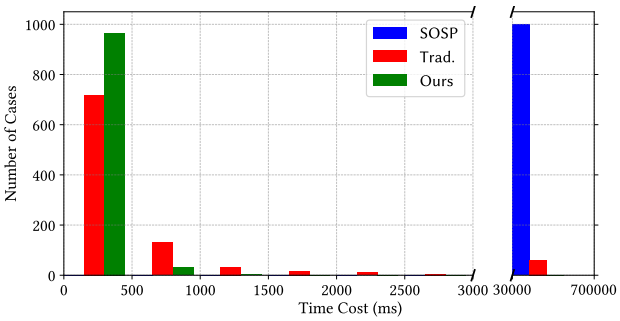


Fig. 4. The distribution of time costs for our method, the traditional inclusion-based method (Trad.) and the SOSP method described in Zhang et al. [2023a] across 1000 random cases. The SOSP is computationally expensive in all the cases. In contrast, our method handles 996 cases within 1 s and takes less than 0.5 s for most problems. The time cost of Trad. varies significantly, with a maximum time consumption of 654 s.

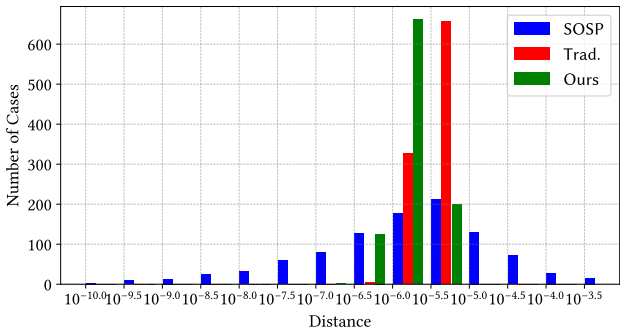


Fig. 5. The distribution of the constraint residuals for our method, the traditional inclusion-based method (Trad.) and the SOSP method described in Zhang et al. [2023a] across 1000 random cases.

(2) The SOSP method described in [Zhang et al. 2023a, §5.2]. The CCD problem is formulated as $\mathrm{CCD}^{2Da}_{d_1,d_2}$, with $[d_1, d_2]$ chosen as $[4, 2]$, $[4, 3]$ and $[6, 4]$ for triangular and quadrilateral patch of order 1, 2, 3 respectively. The code runs on MATLAB R2023b and requires the YALMIP v20230622 and MOSEK 10.1 libraries. Only the runtime of the MOSEK solver is reported, as done in [Zhang et al. 2023a]. The numerical solver employed by the SOSP method to solve the converted SDP does not involve a convergence threshold, so the SOSP method provides no guarantee of solution accuracy.

We test all the three methods using various geometry primitives, including triangular patches and quadrilateral patches of first, second and third order. For each type of geometry primitive, we randomly generate 1000 cases of $\{\boldsymbol{p}_{ij}^{(1)}, \boldsymbol{p}_{ij}^{(2)}, \dot{\boldsymbol{p}}_{ij}^{(1)}, \dot{\boldsymbol{p}}_{ij}^{(2)}\}$ according to a uniform distribution within the range $[-1, 1]$. To further encourage collision occurrences, we also randomly select a direction $\boldsymbol{d}$ from the same distribution, adding $\boldsymbol{d}$ to each vector of $\boldsymbol{p}_{ij}^{(1)}$ and $\dot{\boldsymbol{p}}_{ij}^{(2)}$, and subtracting $\boldsymbol{d}$ from each vector of $\boldsymbol{p}_{ij}^{(2)}$, $\dot{\boldsymbol{p}}_{ij}^{(1)}$.

We report the time costs of the three methods under different settings in Table 1. Although both the traditional method and our method exhibit rapid solving speeds with linear patches, the time consumption of the traditional method increases significantly as accuracy requirements rise(e.g. from 0.2 s to 10 s on third-order quadrilateral patches with OBB-based inclusions when using tolerance of 10^{-4} and 10^{-6}). In contrast, our method shows a more moderate increase in time consumption (from 0.027 s to 0.079 s for tolerance of 10^{-4} and 10^{-6} under the same setting), demonstrating an acceleration factor ranging from 36x to 138x with OBB-based inclusions. Additionally, since OBB gives a tighter description of patches, OBB-based methods outperform AABB-based methods across all the experiment settings. The acceleration effect is particularly evident in time-dependent inclusion-based method, as a tighter description facilitates faster localization and convergence to the collision point. At a precision of 10^{-6}, the SOSP method, when compared with the traditional method, demonstrates much faster solving speeds on second-order surfaces but slower performance on third-order surfaces, as it requires a higher order of SOS polynomials to enhance the accuracy of its solutions. Furthermore, our method achieves a considerable speedup over all surface primitives compared to the SOSP method. Notably, as shown in the second to last column, our method is the only one among the three that consistently achieves a high solving speeding (approximately 0.1 s) for nonlinear patches at a prevision level commonly used in simulations($\delta = 10^{-6}$).

We further conduct an in-depth study of the three methods with the tolerance $\delta = 10^{-6}$ on third-order quadrilateral patches, using OBB as the inclusion function.

Time cost. As shown in the histogram in Fig. 4, we present detailed statistics on the time costs for the three methods. The SOSP is

computationally expensive in all the cases, with time cost exceeding 30 s. In contrast, our method handles 996 cases within 1 s and takes less than 0.5 s for most problems. The maximum time consumption for any case using our method is 2.6 s. The time cost of Trad. varies significantly from case to case, with some cases requiring a large amount of time, the maximum time consumption reaches 654 s.

Worst cases. The most time-consuming cases for the traditional method occur when the two surfaces have non-local manifold contact. The large potential contact region forces the algorithm to extensively subdivide the 5D space across the entire manifold to narrow down to the desired solution, as reported by Snyder et al. [1993]. However, this issue does not arise in our method, as the accurate collision time is determined through algebraic calculations rather than bisection. Cases that are challenging for our approach (taking tens of seconds per case) typically occur when a section of the parameter coordinates maps to a tiny local area in the world space. In these instances, the bounding boxes do not shrink sufficiently with subdivision if extremely high accuracy ($\delta \leq 10^{-8}$) is required. An illustrative example of this scenario is a degenerate patch where different control points locate at the same position. Fortunately, these situations are uncommon in simulation unless intentionally designed.

Constraint residual. To better compare the precision of different methods, we calculate the L2-norm residual of the constraints $|\boldsymbol{F}|_2$, which represents the actual distance between the earliest collided point pair in the world space. Its residual distribution is presented in Fig. 5. As shown, SOSP provides no guarantees on the upper bound of residuals. The residuals of inclusion-based methods align with the magnitude of the precision threshold $\delta = 10^{-6}$ as we set, consistent with our generation rules. In terms of satisfying the constraints, our method generally provides more precise solutions than tradition method under the same tolerance.

Convergence curves. To further demonstrate the accuracy and convergence of our method, we examine the absolute errors between the numerical solutions at different tolerances and the exact solutions. The exact solutions were obtained by solving 1000 cases with a precision of $\delta = 10^{-10}$ using our method, with an average time cost of 5.8 s. In all the 988 cases where collisions occurred, We compute the average errors for $E_t = |\hat{t}^* - t^*|$ and $E_{uv} = |(\hat{u}_1^*, \hat{v}_1^*, \hat{u}_2^*, \hat{v}_2^*) - (u_1^*, v_1^*, u_2^*, v_2^*)|_2$ respectively, where a hat denotes the numerical value of a variable. As shown in Fig. 6, the errors for both metrics decrease at least linearly with the tolerance. Additionally, the numerical solutions obtained using our method provide a closer approximation to the true solution than those from the traditional method under the same tolerance. The solving process of the traditional method with a precision of $\delta = 10^{-7}$ was terminated early due to the excessive memory usage.

5.2 Simulation Examples

To demonstrate the efficacy of our CCD method within real simulation pipelines, we integrate it into an elastodynamic simulator designed for bicubic Hermite patches [Ni et al. 2024]. During collision detection, these patches are equivalently converted into third-order quadrilateral Bézier patches. Upon detecting collisions, the system

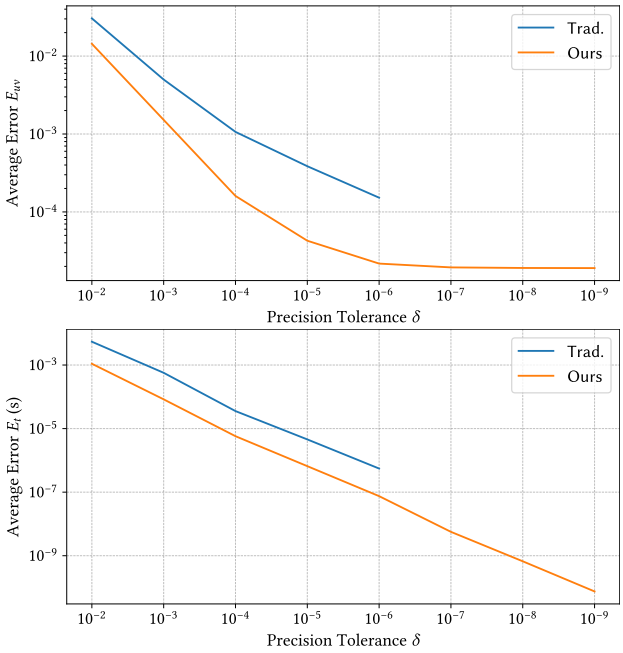


Fig. 6. Convergence curves of the average absolute errors the numerical solutions at different tolerances and the exact solutions for $E_t = |\hat{t}^* - t^*|$ and $E_{uv} = |(\hat{u}_1^*, \hat{v}_1^*, \hat{u}_2^*, \hat{v}_2^*) - (u_1^*, v_1^*, u_2^*, v_2^*)|_2$ respectively.The exact solutions were obtained by solving 1000 cases with a precision of $\delta = 10^{-10}$ using our method.

reverts to the earliest time of impact and updates the velocities of the nodes using an impulse-based solver [Harmon et al. 2008; Ni et al. 2024]. We devised two scenarios to evaluate performance, and the results indicate that our method operates quickly and robustly, delivering efficient and stable outcomes without interpenetration.

Cloth draping on a teapot. As illustrated in Fig. 7, a square sheet of cloth composed of 20×20 patches falls onto a teapot composed of 32 patches under the influence of gravity, eventually sliding down along the handle. As can be seen in the supplementary video, the simulation runs smoothly and stably at a 2 ms time step, with no visible interpenetration. To comprehensively evaluate our CCD method, we select 5 representative frames from the simulation, each corresponding to a subfigure in Figure 7. The time costs for performing CCD detection on these frames using both the traditional method and our method across five commonly used time steps ΔT are reported in Table 2. In the first two frames, 94 and 113, the cloth falls toward the teapot and comes into contact with the spout and the body, resulting in a relatively high relative speed between the two objects. Consequently, the inclusion generated by the traditional method (formulated as the swept volume of the patch) becomes an overly loose estimate, necessitating a large number of subdivision iterations and a long solution time. This drawback is further magnified as the time step increases, leading to a sharp rise in detection time. In contrast, our method uses a more accurate time-dependent inclusion for intersection detection, enabling it to operate efficiently even under high relative speeds and larger time

Fig. 7. A square sheet of square cloth (20 × 20 bicubic Hermite patches) drapes on a teapot (32 bicubic Hermite patches) under gravity, eventually sliding down along the handle. The simulation runs smoothly and stably at a 2 ms time step, with no visible interpenetration between cloth and teapot.

Fig. 8. A square sheet of cloth (10 × 10 bicubic Hermite patches) is pinned at two diagonal corners and drapes under gravity. The pinned points are then moved along the opposite boundaries to the other pair of diagonal corners. Throughout the process, the cloth deforms smoothly with no locking and interpenetration.

steps, thereby achieving significant speed improvements compared to the traditional method. In frames 278 and 356, the cloth's primary movement shifts to sliding along the teapot, leading to a smaller relative velocity along the normal direction of the contact surface, making it easier for the traditional method to resolve. However, our method remains slightly faster. In frame 389, when the cloth and teapot separate, only a few iterations are needed to conclude there is no collision. Therefore, the higher computational cost per iteration of our method makes it slower in this scenario. Nonetheless, this cost is relatively minor compared to other frames and can be largely mitigated through broad-phase culling.

Cloth pinned by two corners. As shown in Fig. 8, a square sheet of cloth consisting of 10×10 patches is suspended under gravity with its two diagonal corners pinned. Once the cloth reaches a stable state, we slide the pinned points along the opposite boundaries to the other pair of diagonal corners. To detect self-collisions within the cloth, we apply our CCD method between non-adjacent patches, while ignoring potential collisions between adjacent patches and within individual patches. As demonstrated in the supplementary video, our method not only manages collisions between these high-order patches generated during patch sagging, but also reliably detects and handles collisions at the boundaries during fixed-point sliding. Unlike sampling-based CCD strategy [Ni et al. 2024], which can miss boundary collisions leading to self-penetration, our approach effectively avoids these issues without requiring the special treatment needed in the method of Snyder et al. [1993].

5.3 Other Examples

Applications on EE tests and VF tests. CCD between linear triangular surfaces is commonly conducted by EE tests and VF tests, both of which are three-variable degenerate cases of CCD between parameter surfaces. We implement both our method and the traditional method for these tests and compared their efficiency and

Table 2. The statistic of time costs for performing CCD detection on the selected frames as shown in Fig. 7 using both the traditional method and our method across five commonly used time steps ΔT.

ΔT	0.001 s	0.002 s	0.005 s	0.01 s	0.02 s
Trad. (OBB)					
Fr. 94	14.43	14.83	157.82	181.02	148.26
Fr. 113	8.94	5.13	4.97	13.77	339.65
Fr. 278	4.30	4.55	4.81	4.91	5.33
Fr. 356	5.80	7.31	9.13	12.32	13.36
Fr. 389	0.04	0.04	0.04	0.04	0.04
Ours. (OBB)					
Fr. 94	1.72	1.73	1.89	1.90	1.99
Fr. 113	1.10	1.10	1.10	1.11	1.33
Fr. 278	2.13	2.16	2.18	2.18	2.19
Fr. 356	1.08	1.08	1.08	1.14	1.36
Fr. 389	0.34	0.29	0.30	0.32	0.31

accuracy on the large-scale benchmark datasets proposed by Wang et al. [2021]. We construct the inclusion functions by calculating bounding boxes rather than directly using interval arithmetic, making our implemented of the traditional method more comparable to the method proposed by Wang et al. [2021] than their implemented IRF method. The convergence tolerance was set to 10^{-6}, as in Wang et al. [2021]. As shown in Table. 3, our method produces no false negatives in any of the cases and maintained a low number of false positives. For the handcrafted dataset, which contains a comprehensive range of collision cases, including various degenerate scenarios and scenarios with multiple solution, our method performs as well as the Trad. method in VF tests and outperforms it in EE tests, where more challenging cases, such as parallel edges, may occur. Our proposed method is the first inclusion-based method to achieve performance that is competitive with other numerical root-finding methods on this dataset. For the simulation dataset,

Table 3. The statistics of the average runtime in μs (t), number of false positive (FP), and number of false negative (FN) for both of our method and the traditional method (Trad.) on the large-scale benchmark datasets proposed by Wang et al. [2021].

	Trad. (AABB)	Ours (AABB)	Trad. (OBB)	Ours (OBB)
Handcrafted Dataset (21K) – Vertex-Face CCD				
t	116.18	67.31	5.44	7.08
FP	41	49	94	59
FN	0	0	0	0
Handcrafted Dataset (34K) – Edge-Edge CCD				
t	387.28	57.27	1097.42	3.85
FP	91	101	300	117
FN	0	0	0	0
Simulation Dataset (18M) – Vertex-Face CCD				
t	0.41	2.17	0.17	1.63
FP	2	2	4	4
FN	0	0	0	0
Simulation Dataset (41M) – Edge-Edge CCD				
t	1.17	4.60	3.66	1.55
FP	9	9	18	38
FN	0	0	0	0

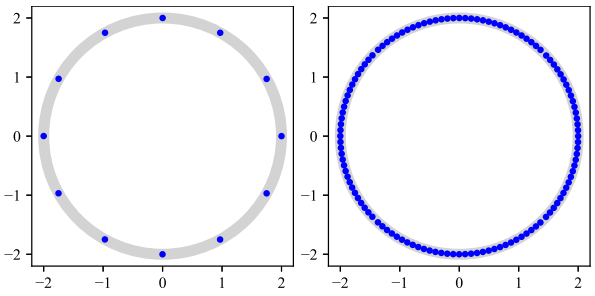


Fig. 9. Solution to the torus-torus CCD problem. We illustrate the true collision manifold as a grey circle in the collision plane, with detected collision points marked in blue. The solution separation distances are set to 1 and 0.1 in the left and right figures, respectively.

our method performs comparably to the Trad. method in EE tests but falls behind in VF tests. This is likely because, in real simulation applications, VF tests are less prone to the same challenges found in handcraft data. Additionally, the simplicity of the calculation and the low dimensionality of the subdivision parameter space contribute to the faster performance of the traditional method.

NURBS patches. (Trimmed) Non-Uniform Rational Bézier Splines (NURBS) surfaces are widely used in CAD applications due to their ability to represent complex shapes with high flexibility and precise local control. A NURBS of order $n \times m$ can be divided into a finite number of rational Bézier patches of same order without any loss of precision through knot insertion [Efremov et al. 2005; Piegl and Tiller 1995]. When the weights assigned to the control points of a rational Bézier surface are non-negative, this surface satisfies the convex-hull property. Concequently, we can apply our proposed CCD framework to rational Bézier patches in a manner similar to polynomial Bézier patches, where the position of control points are derived from the homogeneous coordinates. As shown in Fig. 1, we solve a multi-frame CCD problem involving a bunny composed of 292 linear triangular patches and a torus composed of 16 converted second-order rational Bézier quadrilaterals. During implementation, We approximate their trajectory using piecewise linear motions with a time step of $\Delta T = 0.02$ s. Our method takes 0.66 s to solve the CCD problem in the time step when the two objects come into collision. Our method can not handle rational Bézier patches with negative weights, as the convex hull properties no longer holds in such case. Given that this type of patch can exhibit unusual and non-intuitive behaviors compared to patches with only positive weights, we consider this topic to be beyond the scope of this paper.

Non-isolated point collision manifold. Our method can be integrated into the framework of multi-point collision detection [Snyder et al. 1993] to accurately describe non-isolated point contact manifolds. As shown in Fig. 9, when a torus drops onto another torus below it, a circular contact manifold forms. We set the simultaneity threshold to 1 ms and adjust the solution separation distance to achieve multi-point characterizations with varying density. Within $\Delta T = 1$ s, our method takes 0.49 s to find 12 distinct points with a separation distance of 1, and 1.25 s to find 124 distinct points with a separation distance of 0.1.

6 CONCLUSIONS AND DISCUSSIONS

We propose a time-dependent inclusion-based framework that efficiently and robustly addresses the problem of CCD between parametric surfaces satisfying the convex-hull property and the linear-trajectory assumption. The core of our framework is a newly developed model of inclusion functions, which represent the inclusion of a moving surface as a time-dependent function, along with an integrated intersection detection algorithm. These components provide a precise estimate of the surface trajectory and enable rapid computation of the time period containing potential collisions, thereby eliminating the need for time interval bisection to pinpoint the collision time. Our method demonstrates scalability in meeting increasing accuracy requirements compared to traditional approaches, achieving state-of-the-art performance in CCD problems involving various geometric primitives.

While our framework exhibits zero false negatives in our experiments, its resilience to floating-point errors remains uncertain. To enhance the theoretical safety of the algorithm, we plan to improve our method by incorporating exact geometric computation (EGC) paradigms [Yap 2004], as done for triangle-mesh CCD [Brochu et al. 2012; Tang et al. 2014] and boundary evaluation of curved solids [Keyser et al. 2002], or deriving error bounds via forward error analysis [Wang 2014; Wang et al. 2015]. Additionally, our framework does not accommodate curved trajectories, such as rigid body rotation, where the inclusions intersection detection involves testing nonlinear 2D boundary contours, which exceeds the current capability of our algorithm. promising direction is to adaptively

linearize the curved trajectory and approach the real solution incrementally, referring to the additive CCD method [Li et al. 2021] and its adaptation in rigid body simulation [Ferguson et al. 2021]. Moreover, our framework faces challenges when addressing self-collisions within a single high-order patch and between adjacent patches. To overcome this problem, we intend to investigate continuous self-collision detection scheme tailored to parametric surfaces and integrate them into our framework.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive comments. This work was partially supported by the National Key R&D Program of China 2022ZD0160802. We credit the Houdini Education license for producing the video animations.

REFERENCES

Robert Bridson, Ronald Fedkiw, and John Anderson. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph.* 21, 3 (jul 2002), 594–603. https://doi.org/10.1145/566654.566623

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

R. P. R. Cardoso and O. B. Adetoro. 2017. On contact modelling in isogeometric analysis. *European Journal of Computational Mechanics* 26, 5-6 (2017), 443–472. https://doi.org/10.1080/17797179.2017.1354575

J. Austin Cottrell, Thomas J. R. Hughes, and Yuri Bazilevs. 2009. *Isogeometric Analysis: Toward Integration of CAD and FEA* (1st ed.). Wiley Publishing, Hoboken, NJ, USA.

Octave Crespel, Emile Hohnadel, Thibaut Metivet, and Florence Bertails-Descoubes. 2024. Contact detection between curved fibres: high order makes a difference. *ACM Trans. Graph.* 43, 4, Article 132 (jul 2024), 23 pages. https://doi.org/10.1145/3658191

Alexander Efremov, Vlastimil Havran, and Hans-Peter Seidel. 2005. Robust and numerically stable Bézier clipping method for ray tracing NURBS surfaces. In *Proceedings of the 21st Spring Conference on Computer Graphics* (Budmerice, Slovakia) *(SCCG '05)*. Association for Computing Machinery, New York, NY, USA, 127–135. https://doi.org/10.1145/1090122.1090144

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

Zachary Ferguson, Minchen Li, Teseo Schneider, Francisca Gil-Ureta, Timothy Langlois, Chenfanfu Jiang, Denis Zorin, Danny M. Kaufman, and Daniele Panozzo. 2021. Intersection-free Rigid Body Dynamics. *ACM Transactions on Graphics (SIGGRAPH)* 40, 4, Article 183 (2021).

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.

Stefan Gottschalk. 1996. Separating axis theorem.

David Harmon, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. 2008. Robust treatment of simultaneous collisions. In *ACM SIGGRAPH 2008 Papers* (Los Angeles, California) *(SIGGRAPH '08)*. Association for Computing Machinery, New York, NY, USA, Article 23, 4 pages. https://doi.org/10.1145/1399504.1360622

M. Hughes, C. DiMattia, M.C. Lin, and D. Manocha. 1996. Efficient and accurate interference detection for polynomial deformation. In *Proceedings Computer Animation '96*. 155–166.

T.J.R. Hughes, J.A. Cottrell, and Y. Bazilevs. 2005. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering* 194, 39 (2005), 4135–4195.

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

Minchen Li, Danny M. Kaufman, and Chenfanfu Jiang. 2021. Codimensional Incremental Potential Contact. *ACM Trans. Graph. (SIGGRAPH)* 40, 4, Article 170 (2021).

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

Xingyu Ni, Xuwen Chen, Cheng yu, Bin Wang, and Baoquan Chen. 2024. Simulating Thin Shells by Bicubic Hermite Elements. *Computer-Aided Design* 174 (2024), 103734. https://doi.org/10.1016/j.cad.2024.103734

Les Piegl and Wayne Tiller. 1995. *The NURBS book*. Springer-Verlag, Berlin, Heidelberg.

Xavier Provot. 1997. Collision and self-collision handling in cloth model dedicated to design garments. In *Computer Animation and Simulation*. https://api.semanticscholar.org/CorpusID:7421415

Stéphane Redon, Abderrahmane Kheddar, and Sabine Coquillart. 2002. Fast Continuous Collision Detection between Rigid Bodies. *Computer Graphics Forum* 21, 3 (2002), 279–287. https://doi.org/10.1111/1467-8659.t01-1-00587 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.t01-1-00587

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

M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, and P. Volino. 2005. Collision Detection for Deformable Objects. *Computer Graphics Forum* 24, 1 (2005), 61–81.

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

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

Supplementary: A Time-Dependent Inclusion-Based Method for Continuous Collision Detection between Parametric Surfaces

XUWEN CHEN, School of Intelligence Science and Technology, Peking University, China
CHENG YU, School of Intelligence Science and Technology, Peking University, China
XINGYU NI, School of Computer Science, Peking University, China
MENGYU CHU, State Key Laboratory of General Artificial Intelligence, Peking University, China
BIN WANG*, State Key Laboratory of General Artificial Intelligence, BIGAI, China
BAOQUAN CHEN*, State Key Laboratory of General Artificial Intelligence, Peking University, China

1 INCLUSION INTERSECTION DETECTION

Our time-dependent inclusion-based CCD method is built upon the subdivision framework, where in each iteration we detect potential collisions between two subpatches by conservatively finding the time subinterval when the two time-dependent inclusions of the subpatches intersect during the given time step $[t^{\mathrm{l}}, t^{\mathrm{u}}]$. Specifically, we perform inclusion intersection detection by solving Eq. (18), as described in Alg. 3 in Section 4. It mainly relies on solving an inequality formed as

$$\min_{0\le\alpha<M_1} a_\alpha t + b_\alpha \le \max_{0\le\beta<M_2} c_\beta t + d_\beta, \tag{S1}$$

where on each side the max/min operator is applied on a set of linear functions with respect to t. As illustrated in Fig. 3, we solve this inequality by first compute the convex/concave boundary contour of the max/min function and then find the time subinterval where the min contour exceeds the max contour. We here elaborate our implementation of the algorithm for these two subroutines.

1.1 contour computation

The computation of the max/min contour is indeed to select a subset of the lines that makes up the max/min contour of the original set. We use the index sets $\mathcal{I} \subset \{0, 1, \ldots, M_1 - 1\}$ and $\mathcal{J} \subset \{0, 1, \ldots, M_2 - 1\}$ to indicate the selected lines that forms the contour of the max function and min function, respectively.

Without loss of generality, we focus on the computation of the max contour. As shown in Alg. S1, after sorting and removing the duplicates, we traverse the lines in the original set with the subscript α and store the selected lines in the stack $\mathcal{I}$. When dealing with the α-th line, we (1) first recursively check whether the previously selected line still forms part of the max contour if it were added and (2) then check whether it actually forms part of the max contour and should be added into the stack. Let $\Gamma = \mathrm{SizeOf}(\mathcal{I}) - 1$ so α_Γ denotes the index of the last selected line in $\mathcal{I}$ (the top element in the stack). We formalize the two tests as follows.

ALGORITHM S1: Max contour Computation

Input: A set of lines denoted by slopes and intercepts $\{a_\alpha t + b_\alpha \mid 0 \le \alpha < M_1\}$, the candidate interval $[t^{\mathrm{l}}, t^{\mathrm{u}}]$
Output: An index set $\mathcal{I}$.

```
Sort the lines in ascending order by slope;
Erase lines with same slope but smaller intercept;
Initialize the index set of the selected lines I ← {0};
foreach α do
    do
        if Φ < 0 then
            break
        else
            pop the last index in I
        end
    while I is not empty;
    if I is empty or φ < 0 then
        push α into I
    end
end
return I;
```

Test (1) equals to whether the α-th line intersects with the $\alpha_{\Gamma-1}$-th line after the α_Γ-th line does, as shown in Fig. S1. If so, the α_Γ-th line should be popped out of $\mathcal{I}$. When there is only one line left in the stack, the "$\alpha_{\Gamma-1}$-th line" is chosen as $t - t^{\mathrm{l}} = 0$. So the whole test is interpreted as $\Phi < 0$, where

$$\Phi = \begin{cases} (a_{\alpha_\Gamma} - a_{\alpha_{\Gamma-1}})(b_\alpha - b_{\alpha_{\Gamma-1}}) - (a_\alpha - a_{\alpha_{\Gamma-1}})(b_{\alpha_\Gamma} - b_{\alpha_{\Gamma-1}}), & \Gamma > 0, \quad \text{(S2a)} \\ (a_\alpha - a_{\alpha_\Gamma})t^{\mathrm{l}} + (b_\alpha - b_{\alpha_\Gamma}), & \Gamma = 0. \quad \text{(S2b)} \end{cases}$$

The two cases are depicted in Fig. S1. If the test passes, the Γ-th line is popped and the test continues for the next top line in the stack until the stack is cleared or the test fails.

Test (2) equals to whether the α-th line intersects the α_Γ-th line before t^{u}, as shown in Alg. S2. This test is interpreted as $\phi < 0$, where

$$\phi = (a_{\alpha_\Gamma} - a_\alpha)t^{\mathrm{u}} + (b_{\alpha_\Gamma} - b_\alpha). \tag{S3}$$

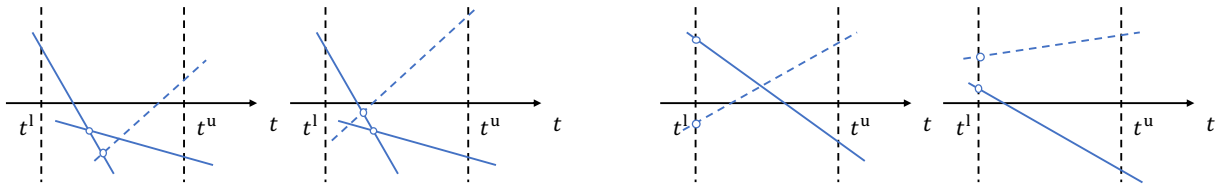
If the test passes, the α-th line is selected as constructing part of the max contour.

We point out that we exclude equal sign in both inequality tests in order to leave out redundant lines.

The computation of the min contour shares a similar process, with a totally reversed sorting and a little revision in computing

*corresponding authors

Authors' addresses: Xuwen Chen, pku_xwchen@163.com, School of Intelligence Science and Technology, Peking University, Beijing, China; Cheng Yu, chengyupku@pku.edu.cn, School of Intelligence Science and Technology, Peking University, Beijing, China; Xingyu Ni, nixy@pku.edu.cn, School of Computer Science, Peking University, Beijing, China; Mengyu Chu, mchu@pku.edu.cn, State Key Laboratory of General Artificial Intelligence, Peking University, China; Bin Wang, binwangbuaa@gmail.com, State Key Laboratory of General Artificial Intelligence, BIGAI, Beijing, China; Baoquan Chen, baoquan@pku.edu.cn, State Key Laboratory of General Artificial Intelligence, Peking University, Beijing, China



case1: $SizeOf(\mathcal{I}) > 1$ case2: $SizeOf(\mathcal{I}) = 1$

Fig. S1. Two different cases of testing whether the top line in the stack should be popped. The solid lines denote the lines in the stack and the dotted line denotes the line being checked. In each case, the test fails in the left figure and passes in the right figure.

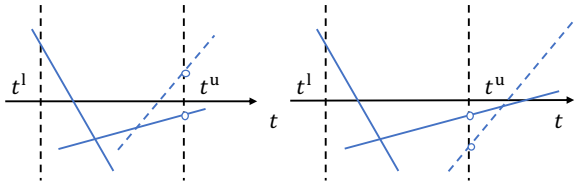


Fig. S2. Testing whether the top line in the stack should be popped. The solid lines denote the lines in the stack and the dotted line denotes the line being checked. The test passes in the left figure and fails in the right figure.

ALGORITHM S2: Left Endpoint of contour Intersection

Input: A max contour denoted by $\{a_{\alpha_\gamma} t + b_{\alpha_\gamma} \mid \alpha_\gamma \in \mathcal{I}\}$, a min contour denoted by $\{c_{\beta_\lambda} t + d_{\beta_\lambda} \mid \beta_\lambda \in \mathcal{J}\}$, the candidate interval $[t^{\mathrm{l}}, t^{\mathrm{u}}]$

Output: The left endpoint of the intersection interval of the two contours

if $\psi < 0$ **then**
  return t^{l};
end
do
  if $a_{\alpha_\gamma} \geq c_{\beta_\lambda}$ **then break**;
  if $\Psi_{\max} < 0$ **then**
    if $\Psi_{\min} < 0$ **then**
      return $-(b_{\alpha_\gamma} - d_{\beta_\lambda})/(a_{\alpha_\gamma} - c_{\beta_\lambda})$;
    else
      $\lambda \leftarrow \lambda + 1$;
    end
  else
    $\gamma \leftarrow \gamma + 1$;
  end
while $\gamma <$`SizeOf` $(\mathcal{I})$ **and** $\lambda <$`SizeOf` $(\mathcal{J})$;
return null;

Φ and ϕ as

$$\Phi = \begin{cases} (a_{\alpha_\Gamma} - a_{\alpha_{\Gamma-1}})(b_\alpha - b_{\alpha_{\Gamma-1}}) - (a_\alpha - a_{\alpha_{\Gamma-1}})(b_{\alpha_\Gamma} - b_{\alpha_{\Gamma-1}}), & \Gamma > 0, \text{(S4a)} \\ -[(a_{\alpha_\Gamma} - a_\alpha)t^{\mathrm{l}} + (b_{\alpha_\Gamma} - b_\alpha)], & \Gamma = 0, \text{(S4b)} \end{cases}$$

$$\phi = -[(a_{\alpha_\Gamma} - a_\alpha)t^{\mathrm{u}} + (b_{\alpha_\Gamma} - b_\alpha)]. \tag{S5}$$

1.2 contour intersection computation

After obtaining the max-contour subset $\{(a_{\alpha_\gamma}, b_{\alpha_\gamma})\}_{\alpha_\gamma \in \mathcal{I}}$ and the min-contour subset $\{(c_{\beta_\lambda}, d_{\beta_\lambda})\}_{\beta_\lambda \in \mathcal{J}}$ sorted by slope in ascending and descending order, respectively, we simultaneously traverse the max-contour and min-contour subsets by traversing the index sets $\mathcal{I}$ and $\mathcal{J}$ using subscripts γ and λ individually. Let $\Gamma = \mathrm{SizeOf}(\mathcal{I}) - 1$ and $\Lambda = \mathrm{SizeOf}(\mathcal{J}) - 1$. Since the max contour is convex and the min contour is concave, the intersection would be null or a single connected interval of t. So next we explain how to find the left end of this interval or achieve the conclusion that intersection does not exist. The process for finding the right end follows the same approach with the traversing order reversed.

As shown in Alg. S2, to find the left end is to find where the two segment contours first intersect. If the max contour starts at a value below the value of the min contour at the start of the candidate time interval t^{l}, satisfying

$$\psi = (a_{\alpha_\gamma} - c_{\beta_\lambda})t^{\mathrm{l}} + (b_{\alpha_\gamma} - d_{\beta_\lambda}) < 0, \tag{S6}$$

we can directly set the left end as t^{l}, otherwise we continue to sweep the contours. If the α_γ-th max-contour line and the β_λ-th min-contour line intersect, the intersection point must (1) lie before the α_γ-th line segment ends and (2) lie before the β_λ-th line segment ends.

Condition (1) is checked by comparing the intersection of the β_λ-th min-contour line with the α_γ-th max-contour line and with the $\alpha_{\gamma+1}$-th max-contour line respectively. "The $\alpha_{\gamma+1}$-th max-contour line" is set as $t - t^{\mathrm{u}} = 0$ when there is no succeeding line after the α_γ-th line. Though there exists three different cases depicted in Fig. S3, condition (1) is satisfied if and only if $\Psi_{\max} < 0$, where

$$\Psi_{\max} = \begin{cases} (a_{\alpha_{\gamma+1}} - c_{\beta_\lambda})(b_{\alpha_\gamma} - d_{\beta_\lambda}) - (a_{\alpha_\gamma} - c_{\beta_\lambda})(b_{\alpha_{\gamma+1}} - d_{\beta_\lambda}), & \gamma < \Gamma, \text{(S7a)} \\ (a_{\alpha_\gamma} - c_{\beta_\lambda})t^{\mathrm{u}} + (b_{\alpha_\gamma} - d_{\beta_\lambda}), & \gamma \geq \Gamma. \text{(S7b)} \end{cases}$$

Condition (2) is satisfied if and only if $\Psi_{\min} < 0$ where

$$\Psi_{\min} = \begin{cases} (a_{\alpha_\gamma} - c_{\beta_{\lambda+1}})(b_{\alpha_\gamma} - d_{\beta_\lambda}) - (a_{\alpha_\gamma} - c_{\beta_\lambda})(b_{\alpha_\gamma} - d_{\beta_{\lambda+1}}), & \lambda < \Lambda, \text{(S8a)} \\ (a_{\alpha_\gamma} - c_{\beta_\lambda})t^{\mathrm{u}} + (b_{\alpha_\gamma} - d_{\beta_\lambda}), & \lambda \geq \Lambda. \text{(S8b)} \end{cases}$$

Once the two conditions are both satisfied, we calculate the left end of the intersection by

$$t = -\frac{b_{\alpha_\gamma} - d_{\beta_\lambda}}{a_{\alpha_\gamma} - c_{\beta_\lambda}}. \tag{S9}$$

If test for condition (1)/(2) fails, i.e., the horizontal coordinate of the intersection point gets beyond the end point of max-contour/min-contour segment, we add γ/λ by one, moving on to the next line

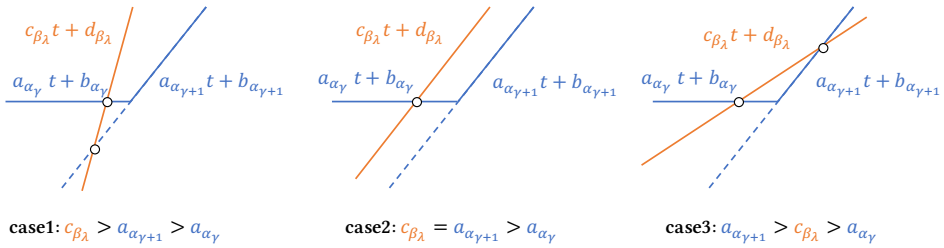


Fig. S3. Three different cases when the γ-th max-contour line and the λ-th min-contour line intersect before the γ-th segment ends.

in the max/min contour. We recursively conduct the tests until c_{β_λ} becomes no larger than a_{α_γ} or we have checked all the lines in either of the two sets before we find an intersection, indicating no intersection happens between the two contours.

If the left end exists, we then compute the right end of the intersection interval using a similar algorithm. The main difference is that the sweep is done from end to start. Each time when condition (1)/(2) fails, we subtract γ/λ by one, moving on to the previous line in the max/min contour. The formalized conditions are written as

$$\psi = (a_{\alpha_\gamma} - c_{\beta_\lambda})t^{\mathrm{u}} + (b_{\alpha_\gamma} - d_{\beta_\lambda}), \tag{S10}$$

$$\Psi_{\max} = \begin{cases} (a_{\alpha_\gamma} - c_{\beta_\lambda})(b_{\alpha_{\gamma-1}} - d_{\beta_\lambda}) - (a_{\alpha_{\gamma-1}} - c_{\beta_\lambda})(b_{\alpha_\gamma} - d_{\beta_\lambda}), & \gamma - 1 \geq 0, \text{(S11a)} \\ (a_{\alpha_\gamma} - c_{\beta_\lambda})t^{\mathrm{u}} + (b_{\alpha_\gamma} - d_{\beta_\lambda}), & \gamma - 1 < 0, \text{(S11b)} \end{cases}$$

$$\Psi_{\min} = \begin{cases} (a_{\alpha_\gamma} - c_{\beta_\lambda})(b_{\alpha_\gamma} - d_{\beta_{\lambda-1}}) - (a_{\alpha_\gamma} - c_{\beta_{\lambda-1}})(b_{\alpha_\gamma} - d_{\beta_\lambda}), & \lambda - 1 \geq 0, \text{(S12a)} \\ (a_{\alpha_\gamma} - c_{\beta_\lambda})t^{\mathrm{u}} + (b_{\alpha_\gamma} - d_{\beta_\lambda}), & \lambda - 1 < 0. \text{(S12b)} \end{cases}$$

1.3 Analysis

The whole algorithm of solving Eq. (S1) for the feasible time interval is at $O(M_1 \log M_1 + M_2 \log M_2)$. Specifically, we compute the max contour by first sorting the line set and then sweeping the set once, which takes $O(M_1 \log M_1)$ and $O(M_1)$ time, respectively. Similarly, the sorting and sweeping during computing the min contour takes $O(M_2 \log M_2)$ and $O(M_2)$ time, respectively. Then we compute the contour intersection by sweeping the two contour subsets simultaneously, which takes $O(M_1 + M_2)$ time. Therefore the bottleneck of time complexity is the sorting process during constructing the max contour and the min contour.

We have tried different ways of constructing the criteria Φs and Ψs, for example, using different intersection points to determine the relationship between the lines, and using the division form of the intersection points without rearranging them into multiplication. All the implementation ways have the same time complexity. We find that such little modifications hardly harm the performance of the overall method. We believe that any reasonable implementation of the sorting-and-sweeping algorithm can work as well as ours.