

Alternating Spatial-Temporal Optimization for Continuous Collision Detection of Signed Distance Fields

Rasmus Gabelgaard Nielsen¹  and Kenny Erleben^{1,2} 

¹ Department of Computer Science, University of Copenhagen, Denmark. ² DARERL ApS, Copenhagen, Denmark.

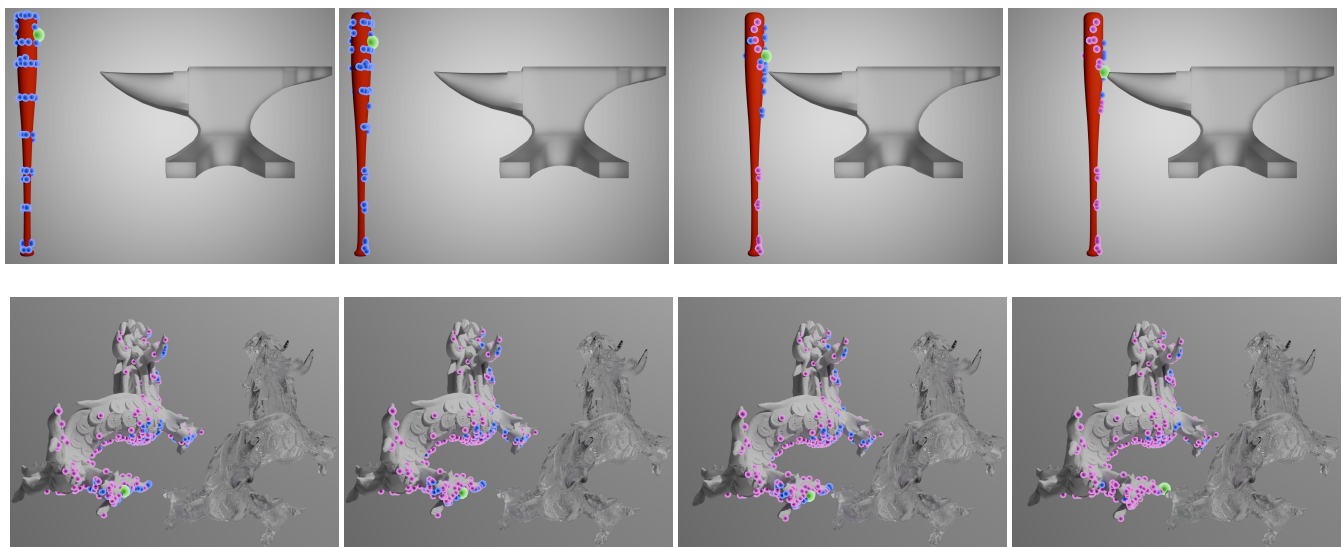


Figure 1: Alternating Spatial-Temporal optimization exploiting smart point sampling results in real-time continuous collision detection between arbitrary complex signed distance fields. From left to right, our numerical method is illustrated: Starting from an initial sparse sampling, it quickly discards non-converging local minima (purple dots), while optimizing for the closest points (blue dots) and selects the earliest time-of-impact among all points (green dot). This results in time of impact computations on the order of 1 ms on a single CPU core.

Abstract

We present a fast, robust method for continuous collision detection (CCD) directly between pairs of complex signed distance fields (SDFs). By casting CCD as an alternating spatio-temporal optimization and driving a small set of smartly sampled surface points along SDF tangent spaces, our approach avoids dense sampling or triangle meshes entirely. The result is real-time SDF-SDF time-of-impact queries—often within 1 ms on a single CPU core—even for highly detailed, non-convex geometry undergoing fast spinning motion. This is, to our knowledge, the first practical CCD method operating directly on SDF pairs.

CCS Concepts

• **Computing methodologies** → **Collision detection**; **Physical simulation**; **Animation**;

1. Introduction

Collision detection is a fundamental component of many computational systems, including physically based simulation, computer graphics, and robotics. Achieving accurate Discrete Collision Detection (DCD) without penetration remains a challenge, as it is inherently prone to missing collisions. Continuous Collision Detec-

tion (CCD) can be employed to either guarantee no false negatives (missed collisions) or, at the very least, drastically minimize them.

Many shape representations exist, such as convex hulls, spheres, or other primitives. While these provide real-time performance, they are unable to accurately represent many shapes. CCD be-

tween triangle meshes, while popular, is challenging to apply in real-time applications if the shape is too detailed. Signed Distance Fields (SDFs), on the other hand, are able to approximate detailed shapes while encoding the signed distance from any point in space to the nearest surface of the shape. This provides several particularly attractive properties, such as the distance value yielding penetration depth and proximity information. Further, the gradient field provides a smoothed surface normal field, which is convenient for defining contact normals for collision resolution, a feat that is tricky with many other representations.

Performing Continuous Collision Detection between a pair of Signed Distance Fields (SDFs) would be advantageous for real-time performance over methods like CCD between triangles, since the CCD problem can be reformulated to depend on the features of the shape, not the number of triangles. We seek a method that, given two SDFs and their motions over a time interval, computes the earliest time of contact for any feature of either SDF. This is quite challenging as determining contact at a given time requires identifying the point in space of closest approach between the SDFs. As the objects move, and especially under rotational motion, the location of this closest point found by a global spatial search can change abruptly, so it is typically non-smooth in time. This is further complicated by the SDF often being non-linear and non-convex. Hence, taking naive or existing SDF approaches and extending them to CCD is impractical, as one would need a rather fine-grained point sampling or high-resolution triangle meshes of their surfaces to not miss any closest points in the time-of-impact computation, and consequently, the search for closest points would be computationally intractable with such a dense fixed surface point sampling. This fundamental barrier is what our approach attacks by taking a different viewpoint of the sampling as a moving point sampling.

To deal with the nonlinearity and complexity of continuous collision detection between two arbitrary signed distance fields, we adopt an optimization process to formulate a generic novel model of the problem. We carefully tailor the model to be agnostic towards an initial sampling by allowing the model to improve on the sample point location during the resolution of the time-of-impact. This is much inspired by Macklin et al. [MEM*20] who applied local-optimization on triangle faces to deal with under-sampling issues in discrete point vs signed distance field collision. Our approach deals with the sampling issue in time as well as relieves us completely from needing triangles. The simplicity of our solution is to derive a mathematical optimization model of the problem where there is no concept of triangles and where sample points are allowed to move tangentially along the zero level set of a signed distance field. We summarize our novel contribution in this work:

- We present a novel optimization model for the time of impact between two SDFs by incorporating a tangent space motion.
- Our tangent space optimization component relieves sampling challenges, and we present a point-SDF representation that aggressively cuts down on the number of needed samples.
- We demonstrate the new approach on highly complex high-resolution examples undergoing both linear motion as well as fast spinning motion, and discuss the numerical optimizations needed for real-time interactive computation.

2. Related Work

General overviews of continuous and discrete collision detection are given by ANDREWS et al. [AEF22] and MIRTICH [Mir96]. These works also cover conservative advancement, collision response, broad- and narrow-phase strategies, and contact generation.

For rigid polyhedral bodies, CCD methods commonly operate directly on triangle meshes. KIM et al. [KR03] approximate relative motion by screw-motion segments and detect vertex/triangle and edge/edge contacts via root finding, later extended by REDON et al. [RKLM04] to articulated models using dynamic bounding-volume culling and swept volumes. ZHANG et al. [ZRLK07] use Taylor models to reduce the number of conservative advancement checks for articulated bodies. BROCHU et al. [BEB12] introduce a geometrically exact CCD for triangle meshes that guarantees to produce correct boolean results (significant collision or not), but is computationally expensive and is unable to achieve real-time performance for many triangles. Deformable geometry introduces additional difficulty due to changing topology. TANG et al. [TKM10] propose a deforming non-penetration filter based on coplanarity conditions, while TANG et al. [TMT10] introduce a local advancement using motion bounds on primitives, avoiding higher-order polynomial solvers. Their method supports self-collisions and arbitrary polygon soups. HARMON et al. [HVTG08] address the complementary problem of collision response under simultaneous contacts.

Given the variety of triangle-based CCD methods available, WANG et al. [WFS*21] collect popular methods into a large-scale benchmark and provide a provably conservative bisection-based algorithm with tight inclusion functions. This avoids false negatives and limits false positives, but is slower than WANG et al. [WTTM15], which trades some false positives for significantly better runtime through Bernstein Sign Classification with floating-point arithmetic. Together, these methods show the recurring trade-off between runtime, false positive rate and even false negative rate that motivates alternative representations. Notably, triangle representations cannot practically achieve real-time performance for the geometric complexity we target, since their cost scales with triangle count rather than the shape's intrinsic features.

Parametric surface representations, such as that proposed by CHEN et al. [CYN*24], can be advantageous for smooth geometry. Here, the convex hull property of parametric surfaces is exploited, and linear motion of control points is assumed over time, yielding an inclusion-based CCD method eliminating the need to subdivide time. For surfaces with many sharp features, however, triangle methods remain competitive, which motivates the SDF method we adopt. Convex representations can be highly efficient; for example, YUAN et al. [YXYS25] show that they can outperform equivalent triangle meshes by a factor of $12\times$, while preserving accuracy by decomposing non-convex meshes into smaller convex meshes. The performance advantage diminishes for more geometrically detailed meshes. Our method is able to achieve similar or even better runtimes, while not being constrained to a convex decomposition. For a restricted set of six specific primitive convex types, TRACY et al. [THM23] achieve performance in the range of $6 - 11 \mu s$, with collision constraints enforceable to machine precision.

Point-SDF methods represent one object as a signed distance field and the other as a point cloud. This is most directly related

to our work. XU et al. [XB17] model each point's motion as a line segment and detect intersection with the SDF by traversing an octree storing per-node minimum and maximum signed distance values, reporting runtimes of 0.7 ms for simple geometry and 9.9 ms for detailed shapes. Their method, however, requires a sufficiently dense point sampling to avoid missing sharp geometric features, a challenge our work addresses by allowing points to move along the surface of the SDF. Our method achieves a $44\times$ speedup on the same geometry. A related approach is taken by PELLETIER-GUÉNÉTTE et al. [PMA25], which represents one body as a triangle mesh and the other as an SDF. Similarly to our approach, they perform a spatio-temporal search constrained to individual triangles to find the earliest time-of-impact at approximately $26\ \mu\text{s}$ per triangle, but are able to discard many triangles via an acceleration structure. In comparison, our method uses around $40\ \mu\text{s}$ for a whole Stanford bunny. Their method achieves an accuracy on the order of 10^{-7} to the true time-of-impact, roughly $100\times$ better than ours. This accuracy is usually required for applications such as haptic rendering, while our accuracy is suitable for game-like applications. The substantial accuracy gap may be attributable to the potential use of analytical SDFs, which can yield significantly more precise distance queries and gradient estimates.

Discrete SDF-SDF collision detection has been explored by LIU et al. [LZW*24], who decompose both SDFs into AABBs, construct shallow octrees at their intersections, and with a gradient method search for maximum penetration depth between the two surfaces. This is, to our knowledge, the only prior work on direct SDF-SDF collision detection, and it serves as an inspiration for our starting point selection strategy.

3. A Spatial-Temporal Optimization Approach

Our idea is to start from a sparse set of sample points restricted to the surface of one body, and perform gradient-based spatio-temporal optimization of their positions to locate the first time-of-impact (TOI) at which both SDFs evaluate to zero.

Let $\mathcal{B}_1$ and $\mathcal{B}_2$ denote a pair of bodies each undergoing motion determined by linear velocity $\vec{v}_c$ and angular velocity $\vec{\omega}_c$ around the center c of the body. For continuous collision detection, we seek the earliest time $\tilde{t} \in [t_{start}, t_{end}]$ at which the two surfaces of $\mathcal{B}_1$ and $\mathcal{B}_2$ touch at any point. We denote this contact point as $x_t \in \mathbb{R}^3$. This must be the earliest point in time when the two surfaces meet. Let $\phi_{\mathcal{B}}(\vec{x}) : \mathbb{R}^3 \rightarrow \mathbb{R}$ denote the signed distance to the surface of the body $\mathcal{B}$. Then the earliest point of contact $\vec{x}_{\tilde{t}}$ must satisfy both $\phi_{\mathcal{B}_1}(\vec{x}_{\tilde{t}}) = 0$ and $\phi_{\mathcal{B}_2}(\vec{x}_{\tilde{t}}) = 0$. Thus, our optimization problem becomes spatio-temporal: Minimize the signed distance of a point $\vec{x}_t$ over time t between both surfaces:

$$\vec{x}_{\tilde{t}}, \tilde{t} \equiv \arg \min_{\vec{x}_t, t} \left(\phi_{\mathcal{B}_1}(\vec{x}_t)^2 + \phi_{\mathcal{B}_2}(\vec{x}_t)^2 \right) \quad (1)$$

$$\text{s.t. } \phi_{\mathcal{B}_1}(\vec{x}_t) \geq 0, \quad (2)$$

$$\phi_{\mathcal{B}_2}(\vec{x}_t) \geq 0, \quad (3)$$

$$t \in [t_{start}, t_{end}], \quad (4)$$

Here, $t_{end} \geq t_{start} \geq 0$. A collision occurs only if the global minimum for t leads to both $\phi_{\mathcal{B}_1}(\vec{x}_t) = 0$ and $\phi_{\mathcal{B}_2}(\vec{x}_t) = 0$. The non-negativity constraint for $\phi_{\mathcal{B}}(\vec{x}_t)$ is important, as squaring the signed

distances makes the objective non-negative while preserving the location of any zero-valued minima. Any strictly positive global minimum implies the trajectories are intersection-free over the interval. Here, the subscript t indicates temporal dependence, as $\vec{x}_t$ is the point evaluated in time t ; thus, the model simultaneously optimizes for the earliest TOI and the spatial location that achieves it.

Solving the aforementioned optimization problem for a single point is challenging: Due to the possibility of non-convex surfaces, gradient-based solvers would struggle to find a global minimum and often converge to a local minimum. Discretizing the surface into points with some uniform spacing, like fast Poisson disk sampling [Bri07], and using those points as starting points for our optimization, would likely have at least one point converge to the global minimum, but as explored by XU et al. [XB17], this approach can be slower, as it requires a sufficiently dense sampling to properly capture all geometric features of the surface. An advantage of this approach is that for a sufficiently dense sampling, the spatial search can be skipped. We instead propose to carefully sample a few points that reduce the risk of converging to a local minimum. This sampling should be much faster than dense sampling, although each point now requires a more expensive spatio-temporal search.

A key design decision is to fix all search points to the surface of $\mathcal{B}_1$ by projecting the points onto it. Enforcing this for all starting points results in the constraint $\phi_{\mathcal{B}_1}(\vec{x}_t) = 0$ being trivially fulfilled. If the bodies are not intersection-free, the global minimum must lie somewhere on $\mathcal{B}_1$; therefore, restricting the spatial search to the surface of $\mathcal{B}_1$ does not exclude true contact solutions. Consequently, we argue that choosing the starting points at the extrema of $\mathcal{B}_1$ only is sufficient to find the point that leads to a solution close to the global minimum, since, as illustrated in Figure 2, choosing too few starting points increases the risk of getting trapped in a local minimum on $\mathcal{B}_1$ due to its local surface features.

3.1. Signed Distance Field Representation

To facilitate spatial-temporal search, we use a Signed Distance Field: Consider a three-dimensional body. Let ϕ be its signed distance field, i.e. the function $\phi : \mathbb{R}^3 \rightarrow \mathbb{R}$ such that $\phi(\vec{x})$ yields the signed distance to the body; note that the set $\{\vec{x} \in \mathbb{R}^3 \mid \phi(\vec{x}) = 0\}$ forms the surface of the body. Further, the gradient of an SDF, $\nabla\phi(\vec{x}) \in \mathbb{R}^3$, satisfies $\|\nabla\phi(\vec{x})\| = 1$; consequently, the surface normal at a point $\vec{x}$ is given by $\vec{n} = \nabla\phi(\vec{x})$. These properties hold exactly for an analytical SDF; on the discrete grid representation we use, $\nabla\phi(\vec{x})$ is approximated by finite differences and carries the same discretization error as the SDF values themselves. This connects directly to our minimization problem: The zero level set defines the constraint surface on which our sample points must lie, and the gradient $\nabla\phi(\vec{x})$ provides the normal used to project points back onto that surface and to compute the tangential spatial step.

Our optimization problem requires frequent evaluations of the signed distance to both SDFs, so we rely on extremely fast SDF queries. Although many SDF representations exist, we intentionally adopt a regular uniform grid representation, being a discrete sampling of the true SDF, rather than more space-efficient structures such as Octrees, OpenVDB or NanoVDB [FPRJ00; Mus13; Mus21]: Since runtime-performance is desired, a regular grid rep-

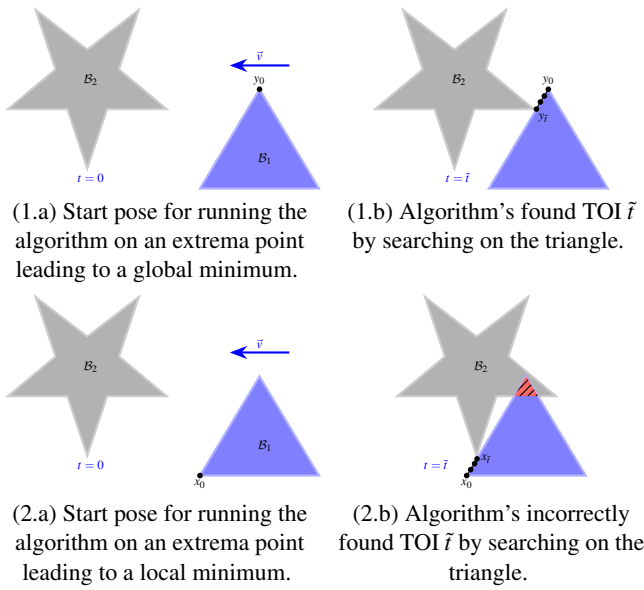


Figure 2: In (1.b), the algorithm uses the correct starting point selected at one of the extrema. This leads to the algorithm finding the global minimum. In (2.a), a starting point at a different extremum is used. Observe how in (2.b), the spatio-temporal search for the selected extrema point can get stuck at one feature of B_2 , thus converging to the wrong local minimum, leading to penetration.

representation is absolutely unparalleled in terms of lookup, as a grid-lookup is an index-computation followed by a single array access (with some trilinear interpolation if our sample point does not coincide with a point in the regular grid). However, this choice comes at the cost of memory usage, which rapidly increases when the grid resolution increases, where more efficient structures can reduce storage by one to three orders of magnitude. We did not explore the use of neural representations of SDFs as in [PFS*19], since their inference costs are anticipated to impede real-time performance; our proof-of-concept implementation therefore uses regular grids. However, neural representations remain a promising avenue to explore, as they provide significant benefits, such as substantially reducing memory usage, providing more accurate gradients, and continuous signed distance point queries.

Further, we chose not to use analytical SDFs due to their inability to easily represent arbitrary triangle meshes. While our proof-of-concept implementation depends on discretized grids, our method is easily adaptable to any of the discussed SDF representations.

3.2. Smart Point Sampling

Since each point will be subjected to a spatial search, convergence typically occurs at geometric extrema of the surface. In particular, when contact generation is required, failing to include starting points in all relevant extrema may result in missed contacts. We therefore design a sampling strategy that targets extremal surface features, while working for geometries without clearly distinguished "peaks" or "valleys". To facilitate this process, we use a discretized grid: Here, detection of extrema is simple. If the SDF is represented analytically, neurally or similar, we suggest temporar-

ily constructing a discretized grid (e.g., octree-based), to be used solely for generating starting points. These initial points are computed once as a preprocessing step and reused throughout the simulation. Extrema are identified directly on the discretized grid. A grid point is classified as a local maximum if its value is strictly greater, along at least one coordinate axis, than the values of its neighbouring samples. Note that all neighbouring points and the grid point are projected onto the closest point on the isosurface. Similarly, a point is declared a local minimum if it is strictly smaller than its neighbours along at least one axis. Selecting only local maxima and minima is insufficient for shapes such as a flat plane, which do not exhibit any extrema under the above criteria. To ensure adequate point coverage in such cases, we expand our sampling procedure:

- Subdivide the SDF into N^3 uniformly sized voxels. Retain only those voxels that intersect with the isosurface and discard the remainder.
- Analyze the surface of the SDF and select points based on the following criteria:
 - Identify all extrema on the isosurface and select starting points at the extrema. These are the points that are most likely to collide with the other SDF.
 - If no extrema are found within a voxel based on the above criteria, select the center of the voxel and project that point to the closest point on the isosurface.
- Finally, to ensure there is a minimum distance between points, those closer than a threshold are pruned, as they would likely converge to the same time-of-impact and therefore provide redundant searches. The discrete nature of the grid can provide some noise, leading to multiple points selected in close proximity, making this step a requirement.

A contact cannot be missed when the sampling covers the full isosurface of the discretized SDF. Complete isosurface coverage is, however, stronger than necessary: Flat regions need only one starting point, since gradient descent from any position on a flat region converges to the same local minimum. Similarly, a sphere requires only two starting points, one at each pole, as these are the only locations where descent can reach a local minimum. The extrema-based strategy is a sparse approximation to this exhaustive sampling, one that works well empirically, but the reliability argument rests on surface coverage rather than extrema alone. The number of samples therefore provides a direct quality-performance trade-off: More samples improve robustness, while fewer reduce runtime cost. Should a miss for our extrema-based sampling, however unlikely, actually occur, it would be difficult to detect at runtime, as the case in which no collision occurs in the search interval, and one in which a collision occurs but all starting points converged to non-contacting local minima looks identical.

For completeness, we list the time it takes to preprocess the starting points in Table 1. This is not part of the runtime cost, and can be pre-baked for any real-life application. Hence, we have not sought to optimize this part of the work.

3.3. The Numerical Optimization Method

Restricting the spatial point search to B_1 offers several advantages: If the SDFs undergo large angular motion, non-projected spatial

Grid resolution	64 ³	128 ³	256 ³	512 ³
XYZ-Dragon	6.11847 ms	49.0732 ms	420.896 ms	3187.86 ms

Table 1: Total computation time for finding extrema on the SDF for grid sizes. Observe linear scaling with resolution.

points could move too quickly away from $\mathcal{B}_1$ towards $\mathcal{B}_2$, leading to empty regions on $\mathcal{B}_1$, which could be where the global minimum lies. Instead, even with a large spin, constraining the points ensures they follow the motion of $\mathcal{B}_1$, guaranteeing the points do not move too far off-course. It also serves as a simplification to the optimization problem. By always projecting the spatio-temporal search point $\vec{x}_t$ to the surface of $\mathcal{B}_1$, we ensure that $\phi_{\mathcal{B}_1}(\vec{x}_t) = 0$. For SDFs, the projection is based on the gradient and distance to the SDF:

$$\vec{x}_{t_{proj}} \leftarrow \vec{x}_t - \phi_{\mathcal{B}_1}(\vec{x}_t) \nabla \phi_{\mathcal{B}_1}(\vec{x}_t). \quad (5)$$

Each point should move towards the closest feature of $\mathcal{B}_2$, while sticking on $\mathcal{B}_1$. While it is possible to have the spatial step move in $\mathcal{B}_2$'s gradient direction, the projection can prevent the point from making any progress. Thus, in the spatial direction, the point moves towards the closest point on the surface of $\mathcal{B}_2$ while moving along the tangential plane of $\mathcal{B}_1$. The temporal direction is based on how quickly $\vec{x}_t$ is moving towards $\mathcal{B}_2$:

$$\vec{d}(\vec{x}_t) = \left[\nabla \phi_{\mathcal{B}_2}(\vec{x}_t) - \frac{\nabla \phi_{\mathcal{B}_2}(\vec{x}_t) \cdot \nabla \phi_{\mathcal{B}_1}(\vec{x}_t)}{\|\nabla \phi_{\mathcal{B}_1}(\vec{x}_t)\|^2} \nabla \phi_{\mathcal{B}_1}(\vec{x}_t) \right] \quad (6)$$

Here, $\vec{v}_t$ is the velocity, which depends on the simulation type. For rigid body simulations, like ours, the instantaneous velocity for the point on a rigid body whose center is at C , with a linear and angular velocity $\vec{v}_C$ and $\vec{\omega}$, is:

$$\vec{v}_t = \vec{v}_C + \vec{\omega}_C \times (\vec{x}_t - C) \quad (7)$$

However, this is under the assumption that only $\mathcal{B}_1$ is moving. If both are moving, we compute the relative velocity:

$$\vec{v}_{rel} = (\vec{v}_{\mathcal{B}_1} + \vec{\omega}_{\mathcal{B}_1} \times (\vec{x}_t - C_{\mathcal{B}_1})) - (\vec{v}_{\mathcal{B}_2} + \vec{\omega}_{\mathcal{B}_2} \times (\vec{x}_t - C_{\mathcal{B}_2})), \quad (8)$$

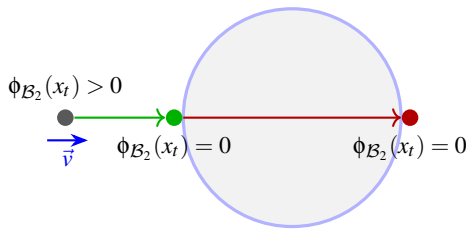


Figure 3: A point moving with a linear velocity. Note how there are two possible positions the point can take in its trajectory, where the distance to the isosurface is exactly zero, but the first point is the only correct choice. Using backtracking line search risks taking overly large steps initially, ending up on the wrong side of the sphere, as visualized with the red point.

Optimizing the objective function in Equation 1 presents a subtle problem: As visualized in Figure 3, given a sample point moving in time, there can be multiple local minima in which both $\mathcal{B}_1$

and $\mathcal{B}_2$ yield a signed distance of zero, while lying on different sides of the surface. This ambiguity undermines standard adaptive step size strategies, such as backtracking line-search. Usually, those start with a large step size value until the Armijo conditions are satisfied; however, the first large step size might cross to the wrong side of the SDF, yet still cause the line search to accept this step, since it appears the point is moving closer to the objective.

We employ projected gradient descent to optimize the objective function: Gradients are cheap to compute, and must be sampled multiple times per search point. Further, it is easy to enforce the constraint of having points projected to the surface of $\mathcal{B}_1$. However, with a constant temporal step size α , too high a value risks overestimating the TOI, while too low a value can lead to slow convergence. A step size α that leads to an accurate TOI would require unacceptably many iterations. To mitigate this, we multiply α by $\phi_{\mathcal{B}_2}$, as this acts as an adaptive step size: When the bodies are far away, the temporal step will be large, while close bodies would take tiny steps. The temporal step size, scaled by $\phi_{\mathcal{B}_2}$ is bounded by the distance to the isosurface, which avoids divergence in practice.

We provide Algorithm 1 for performing the gradient descent. When running the algorithm on each point, we keep track of the best time-of-impact found so far. This is then fed to subsequent points, and if a point's TOI exceeds the current best, we discard it. This halved the runtime without affecting accuracy. Note *simulateVertexPosTo*(x, t) transforms point x from the body's pose at t_{start} to its pose at time t , using the body's linear and angular velocity.

ALGORITHM 1: Gradient Descent SDF-SDF

Input: Time range t_{start} and t_{end} , starting point x_0 , step size α ,

temporal step size β , best time of impact t_{best} found for any point so far.

```

1:  $t_0 \leftarrow t_{start}$ 
2:  $x_{t_0}^i \leftarrow \text{simulateVertexPosTo}(x_0^0, t_0)$ 
3: for  $i = 1$  to max iterations do
4:   if  $t_i > t_{best}$  or  $t_i > t_{end}$  then
5:     discard point
6:   end if
7:   Compute  $v_{t_i}$  as Equation 7 or 8
8:    $\Delta x \leftarrow \nabla \phi_{\mathcal{B}_2}(x_{t_i}^i) - \frac{\nabla \phi_{\mathcal{B}_2}(x_{t_i}^i) \cdot \nabla \phi_{\mathcal{B}_1}(x_{t_i}^i)}{\|\nabla \phi_{\mathcal{B}_1}(x_{t_i}^i)\|^2} \nabla \phi_{\mathcal{B}_1}(x_{t_i}^i)$ 
9:    $x_{t_i}^{i+1} \leftarrow x_{t_i}^i - \alpha \cdot \Delta x$ 
10:   $\Delta t \leftarrow \nabla \phi_{\mathcal{B}_2}(x_{t_i}^{i+1}) \cdot v_{t_i} \phi_{\mathcal{B}_2}(x_{t_i}^{i+1})$ 
11:   $t_{i+1} \leftarrow \text{clamp}(t_i + \beta \cdot \Delta t, t_{start}, t_{end})$ 
12:   $x_{t_{i+1}}^{i+1} \leftarrow \text{simulateVertexPosTo}(x_{t_i}^{i+1}, t_{i+1})$ 
13:   $x_{t_{i+1}}^{i+1} \leftarrow x_{t_{i+1}}^{i+1} - \phi_{\mathcal{B}_1}(x_{t_{i+1}}^{i+1}) \nabla \phi_{\mathcal{B}_1}(x_{t_{i+1}}^{i+1})$ 
14:  if  $\phi_{\mathcal{B}_2}(x_{t_{i+1}}^{i+1}) \leq \epsilon$  then
15:    break
16:  end if
17:  if  $|(t_{i+1} - t_i)| + |(t_{i+1} - t_{i-1})| \leq \epsilon$  then
18:    discard point
19:  end if
20: end for
```

3.4. Optimizations Resulting in a Threefold Speedup

To optimize the algorithm, we employ several strategies, which, when combined, yield an average speedup of $3 \times$:

1. We employ the conservative advancement discussed in [Mir96] to create a tight upper bound for how far in time our body can travel based on its linear and angular velocity and its bounding sphere. It thus yields the earliest possible time two bodies are able to meet. This method guarantees no false negatives, but will, in almost all cases, provide a false positive. However, this is well suited to our case: Computing the earliest possible time is a natural starting time for all points used in our algorithm. This reduces the time range necessary to search for all points.
2. We employ an inexpensive search to obtain the point most likely to hit the SDF first: Compute $\phi_{\mathcal{B}_2}(x_0)$ for all points, then select the point that yields the closest distance to the isosurface, and use its computed time-of-impact as an upper bound (the end of the time range) for all remaining extrema points. If angular velocity is present, the points selected should be the one most likely to reach $\mathcal{B}_2$ first during angular motion, which can be computed based on $\frac{\phi_{\mathcal{B}_2}(\vec{x}_i)}{\nabla \phi_{\mathcal{B}_2}(\vec{x}_i) \cdot \vec{v}_i}$.
3. While processing each point sequentially, keep track of the earliest TOI computed so far, then always use the best TOI found as the upper bound in the TOI-search for all subsequent points.

We initially designed the method to evaluate both orderings of the pair $\mathcal{B}_1$ and $\mathcal{B}_2$. After computing the TOI for the original ordering, $\mathcal{B}_1$ and $\mathcal{B}_2$ were swapped, and the TOI was computed using starting points for $\mathcal{B}_2$. However, this additional step did not yield any meaningful improvement in accuracy and thus was left out.

3.5. Re-using starting points for Contact Generation

Our point-search method offers an additional advantage: Because we hypothesize that the number of contacts needed is limited to the number of extrema, all starting points converging to the earliest time-of-impact can be stored and reused directly as a multi-contact set. The voxelisation we proposed is vital for this, as otherwise the number of available contact points may be insufficient; for example, when contact happens between two perfectly planar surfaces. Contact generation is cheap, incurring only negligible overhead to compute the contact normal, and the contact point and penetration depth are already available from the CCD query.

4. Results

To the best of our knowledge, no other CCD method for SDF pairs has been reported. The closest related work found is a point-SDF approach by XU et al. [XB17]. Like our method, it uses randomized testing and reports results on the Stanford Dragon and Bunny. The reported runtimes are: Bunny: 0.7 ms and Dragon: 9.9 ms. The work does not provide a quantifiable error analysis measuring how the computed CCD queries approximate the true time-of-impact. Their method does, however, compute an exact continuous test between a trilinear interpolated implicit function and a line segment. We therefore expect it to be more accurate than ours, although some of the loss in our method is an inherent consequence of discretizing the SDF grid, particularly in the gradient approximation.

All benchmarks were run on a Linux laptop with an 8-core 16-thread Ryzen 7735HS CPU and 20.5 MB cache. We only use one core. We excluded benchmarks involving multiple pairs and restricted our experiments to pairwise testing. Although broad-phase

strategies such as spatial hashing or temporal BVHs are important in a full CCD pipeline, they are not the focus of this work and would only add noise to the benchmarks. Instead, we focus on non-convex geometry with high geometric complexity and sharp features, using models with equivalently high triangle counts, as these both expose limitations and capabilities of our method on challenging geometry. Further, all meshes are scaled such that they are 1 meter wide. Points closer than 5% of the geometry-size to other points are pruned, and a voxel size of 5^3 are used for all experiments.

ΔTOI	Min	Max	Mean	Avg. Runtime
Dragon	$-7.53 \cdot 10^{-5}$	$9.06 \cdot 10^{-4}$	$6.96 \cdot 10^{-5}$	0.225 ms
Armadillo	$-1.12 \cdot 10^{-3}$	$7.66 \cdot 10^{-3}$	$9.53 \cdot 10^{-5}$	0.203 ms
XYZ-Dragon	$-1.33 \cdot 10^{-3}$	$1.52 \cdot 10^{-3}$	$4.93 \cdot 10^{-5}$	0.277 ms
Bunny	$-5.04 \cdot 10^{-4}$	$1.03 \cdot 10^{-3}$	$7.21 \cdot 10^{-5}$	0.040 ms

Table 2: 2000 shape pairs (Dragon, Armadillo, XYZ-Dragon, and Bunny) with a static shape in the center. We benchmark our Gradient Descent method for SDF-SDF CCD. Ground truth is established by simulating in $\Delta t = 10^{-6}$ steps and using finer steps when contact is detected. We measure mean, min, and max differences between the ground truth and our algorithm's computed time-of-impact, as well as average runtime. Accuracy is close to ground truth across all bodies despite varying geometric complexity.

4.1. Accuracy of CCD-SDF

Most tests are randomized: For each test, we generate 2000 pairs of the same specified shapes (e.g., dragon, armadillo, and others), with randomized positions, orientations and velocity. The linear velocity has a magnitude (such as 100m/s), split randomly into the components of the vector. The ground truth of when the pairs collide is compared with the time-of-impact returned by our algorithm. To obtain reliable results, we choose the random configurations such that most pairs collide within the prescribed time interval.

ΔTOI	Min	Max	Mean	Runtime
Float32	$-1.32 \cdot 10^{-3}$	$1.72 \cdot 10^{-3}$	$5.41 \cdot 10^{-5}$	0.241 ms
Float64	$-1.33 \cdot 10^{-3}$	$1.52 \cdot 10^{-3}$	$4.93 \cdot 10^{-5}$	0.277 ms

Table 3: Comparison of single- and double-precision arithmetic with respect to closeness to the ground-truth TOI for 2,000 seeded pairs of dragon shapes randomly propagated toward a static SDF. Observe how the accuracy only decreases slightly, while the runtime improves. Here the grid discretization error of gradients and gradient descent approximation errors dominate.

Ground truth is computed from the isosurface of the grid: We collect all discrete points on the isosurface, then advance both bodies in small increments $\Delta t = 10^{-6}$ according to their angular and linear velocity. At each step, all isosurface points of one body are sampled against the other SDF to test for penetration. If penetration occurs for any one of the points, we restart the search from the last non-penetrating configuration with time-steps of 10^{-9} . This process is refined once more with steps of 10^{-12} to obtain a more precise estimate of the contact time. It should be noted that this ground truth is itself an approximation: It is based on the isosurface of the discretized SDF, not the true analytical time-of-impact. The reported errors are therefore relative to this approximation.

Since we extract and test all isosurface points, runtime must be measured separately from accuracy evaluation against ground truth; otherwise most of the grid would be cached, making runtime appear significantly better than it is. The randomized tests use only Stanford meshes: Bunny, Dragon, XYZ-Dragon, and Armadillo, with a maximum of 100 iterations and all optimizations from Subsection 3.4. We use the interval $t_{\text{start}} = 0$ and $t_{\text{end}} = 1$, a step-size- $\alpha = 0.1$, and a temporal step-size of 1.0. In Tables 2, 3, and 4, all bodies are assigned random linear velocities with a magnitude of 100 m/s, no angular velocity, and randomized start configurations.

Unless otherwise noted, all tables use 64-bit precision. However, as shown in Table 3, runtime can be improved slightly at lower precision with only a tiny loss in accuracy. This is a surprisingly small difference: We hypothesize that the dominant source of error is the gradient approximation on the discretized grid and the gradient descent error, rather than floating-point precision. Using 32-bit precision is therefore attractive, as it halves the memory footprint, provides a modest speedup of 1.15x, and is better suited for GPU implementations. The trade-off is a slight decrease in accuracy. Table 2 reports the accuracy of the method on a range of shapes with varying geometric complexity. The bunny, being the simplest shape, requires only a small number of starting points and therefore achieves a significantly lower runtime than more complex models.

4.2. Scalability of points

We hypothesize that our method's runtime is dependent on the number of extrema the shapes contain, rather than the number of triangles or the SDF resolution. While our method's runtime is dependent on the number of points, due to all optimizations employed, the runtime should scale more slowly than the number of points used. We show in Table 4 that the number of triangles is not as important as the number of points, and that many high triangle-counts can be represented using significantly fewer points than vertices.

Scalability	Total triangles	Points used	Runtime	Avg. Iterations
Dragon	871414	380	0.225 ms	2.14
Armadillo	345944	388	0.203 ms	1.87
XYZ-Dragon	7218906	594	0.277 ms	2.37
Bunny	1500	43	0.040 ms	4.23

Table 4: Scalability of points in terms of execution time for the entire algorithm, averaged over 2000 shape pairs, randomly shot towards a static SDF. Observe how fewer points often lead to improved runtime. The number of starting points does not necessarily scale directly with the number of triangle, due to the adaptive sampling strategy of the approach. As a result, performance is more dependent on the shape than the representation itself. This separates the visual representation, such as a triangle mesh, from the collision detection discretization, being point samples and SDFs.

4.3. Angular vs Linear Velocity

For this experiment, the step-size- β is 1.1 to avoid using too many iterations while still preserving accuracy. The maximum number of iterations is increased to 200, since a few points may struggle to find a time-of-impact when angular velocity is present.

Studying the effect of angular velocity on our method is important, since points travel based on the gradients of both SDFs. When bodies spin significantly, their poses at the initial guess t_0 and true time-of-impact $\tilde{t}$ may differ substantially. As a result, the points may initially take a step in an incorrect direction. Near the actual TOI, this can require additional steps to travel to the solution, and we therefore expect a higher iteration count, and consequently, a worse average runtime. In practice, however, large angular velocities are uncommon. We therefore separate the effects of linear and angular motion by reporting the linear velocity on its own.

Table 5 shows that the method takes significantly longer when angular velocity is present than when it is absent. Observe how the magnitude of the velocity scales with the proximity to the solution. Here, higher velocity values correspond to a closer approximation to ground truth. For linear velocity alone, we observe a slight speedup once the search interval shrinks.

Linear (m/s)	Angular (rad/s)	Min	Max	Mean	Runtime
15	15	$-4.29 \cdot 10^{-3}$	$7.55 \cdot 10^{-3}$	$9.41 \cdot 10^{-4}$	0.861 ms
150	150	$-2.64 \cdot 10^{-3}$	$6.03 \cdot 10^{-4}$	$4.19 \cdot 10^{-5}$	0.824 ms
1500	1500	$-3.83 \cdot 10^{-4}$	$1.37 \cdot 10^{-4}$	$8.76 \cdot 10^{-7}$	0.941 ms
15000	15000	$-8.45 \cdot 10^{-5}$	$1.00 \cdot 10^{-5}$	$-4.56 \cdot 10^{-6}$	0.939 ms
15	0	$-9.44 \cdot 10^{-3}$	$6.76 \cdot 10^{-3}$	$1.54 \cdot 10^{-4}$	0.277 ms
150	0	$-8.37 \cdot 10^{-4}$	$3.71 \cdot 10^{-4}$	$1.68 \cdot 10^{-4}$	0.263 ms
1500	0	$-8.55 \cdot 10^{-4}$	$3.51 \cdot 10^{-5}$	$-2.83 \cdot 10^{-5}$	0.257 ms
15000	0	$-8.65 \cdot 10^{-5}$	$2.61 \cdot 10^{-6}$	$-7.84 \cdot 10^{-6}$	0.249 ms

Table 5: Scalability of points in terms of linear and angular velocity, with 2000 random XYZ Dragon-XYZ Dragon pairs. We measure mean, minimum, and maximum differences between the ground-truth and our algorithm's computed time-of-impact. Observe that when high angular velocity is present, the algorithm's average runtime increases, compared to the case with only linear velocity.

4.4. Effect of grid resolution on accuracy and runtime

Since the computed gradient is obtained using a finite-difference approximation, increasing the grid resolution is expected to improve the accuracy of the solution, but worsen execution time, as the likelihood of points being cached from previous runs' iterations decreases. Further, finer details in the mesh may emerge, increasing the likelihood of reaching a local minimum. We wish to investigate the importance of an accurate gradient for the algorithm. In these tests, all bodies have the same linear velocity of 50 m/s and no angular velocity, but random start configurations.

Table 6 shows the runtime increases with the grid size. Our CPU has 20 MB cache. Here, 128^3 grid uses 16 MB and fits in cache, unlike the 256^3 , which uses 128 MB. However, the runtime only worsens slightly, likely due to the outer layers of the grid being sampled significantly more; consequently, most samples are in cache. A 512^3 grid uses 1 GB, so cache misses are frequent, worsening runtime performance.

4.5. Convergence Behaviour

We aim to provide a fair characterization of the convergence behaviour of our algorithm. Following the same randomized benchmarking strategy used in the previous experiments, we perform 100 runs and report two separate convergence measures.

Grid Size	Min	Max	Mean	Runtime
64^3	$-2.49 \cdot 10^{-3}$	$9.78 \cdot 10^{-4}$	$-1.27 \cdot 10^{-5}$	0.193 ms
128^3	$-2.75 \cdot 10^{-3}$	$5.94 \cdot 10^{-3}$	$9.98 \cdot 10^{-5}$	0.303 ms
256^3	$-6.68 \cdot 10^{-3}$	$1.49 \cdot 10^{-3}$	$3.59 \cdot 10^{-5}$	0.325 ms
512^3	$-6.67 \cdot 10^{-3}$	$8.95 \cdot 10^{-4}$	$2.93 \cdot 10^{-5}$	0.386 ms

Table 6: Performance of the same XYZ Dragon-XYZ Dragon SDF pair for different grid sizes, compared to our ground truth method. We measure mean, minimum, and maximum differences between the ground-truth and our algorithm's computed time-of-impact. Observe how the runtime is not affected much when we increase the resolution, but the accuracy improves. This is unsurprising as most query points lie outside the discretized domain of the SDF, so the points on the outer layers are accessed repeatedly, keeping this relatively small portion of the grid in the CPU cache.

First, we compare the predicted time-of-impact against the ground truth time-of-impact at each iteration of starting points until convergence. For each of the randomized runs, we only select the starting point that led to our method's earliest predicted TOI and discard all other starting points. This avoids including noisy starting points arising from early exit or worse TOI estimations. The plot should show how quickly the method approaches a nearby solution, with most iterations being small refinements. Figure 4 shows that, in most cases, the method converges rapidly to a solution close to the ground truth, typically within 5-7 iterations.

We further evaluate algorithmic convergence by tracking the starting points, until they satisfy $\phi_{B_2}(x_{t_i}) < \epsilon$. We exclude starting points that terminate via early exit, or exhibit no temporal progress. The remaining points from all runs are collected into a convergence plot in Figure 5. Here we see the error decreases at a similar rate as in Figure 4, although some cases require substantially more iterations. Nevertheless, most points converge in only a few steps before reaching a point sufficiently close to SDF B_2 .

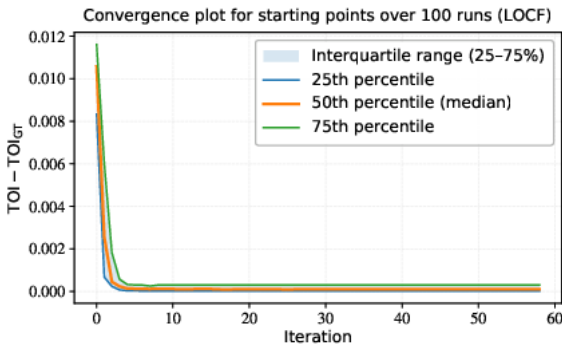


Figure 4: Quartile convergence plot for 100 runs. For each run, convergence is measured using the starting point that yields the earliest predicted time-of-impact. Notice how the algorithm rapidly finds a TOI very close to the true TOI, before converging.

4.6. Ablation study

Our previous benchmarks established that extrema-point sampling yields reasonable time-of-impact estimates. However, we have not

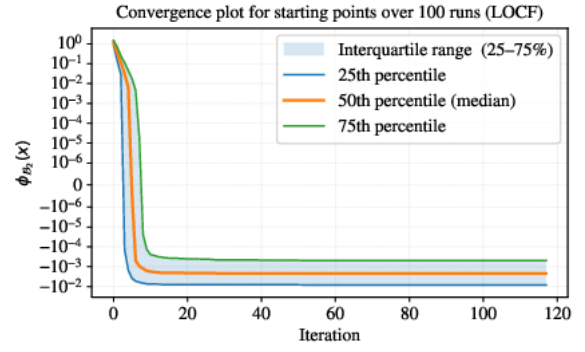


Figure 5: Quartile convergence plot for 100 runs. For each run, convergence is measured over all starting points satisfying $\phi_{B_2}(x_{t_i}) < \epsilon$. Observe, how convergence is mostly achieved within 10 iterations, although some points uses substantially more.

yet experimentally assessed whether it is necessary to sample all extrema points. To investigate this, we reduce the number of points until the method clearly fails. We expect some loss in accuracy, but since starting points may travel far from their initial positions, removing only a few points should rarely affect the estimated TOI.

Points	Min	Max	Mean	Runtime
594	$-1.33 \cdot 10^{-3}$	$1.52 \cdot 10^{-3}$	$4.93 \cdot 10^{-5}$	0.277 ms
367	$-7.21 \cdot 10^{-3}$	$2.65 \cdot 10^{-3}$	$6.32 \cdot 10^{-5}$	0.231 ms
155	$-8.34 \cdot 10^{-3}$	$4.82 \cdot 10^{-3}$	$2.69 \cdot 10^{-4}$	0.079 ms
31	$-6.09 \cdot 10^{-3}$	$7.46 \cdot 10^{-3}$	$9.35 \cdot 10^{-4}$	0.021 ms
1	$-1.36 \cdot 10^{-3}$	$8.66 \cdot 10^{-1}$	$2.39 \cdot 10^{-2}$	0.004 ms

Table 7: Performance of 1000 random seeded XYZ Dragon-XYZ Dragon SDF pairs for a decreasing amount of starting points. We compare our methods' computed time-of-impact with our ground truth method. Min and max denote the minimum and maximum differences between our method and the ground truth across all runs, while mean denotes the average difference. Observe how the accuracy decreases when the amount of points are only reduced slightly.

We use the XYZ-dragon, containing the largest number of extrema in our test set, and use a linear velocity magnitude of 100m/s. In Table 7, we evaluate the method using the same seeded 1000 randomized start configurations. The results show that the method remains surprisingly accurate even as the number of starting points is reduced significantly. Although accuracy decreases as points are removed, it could be considered usable in some applications. When one starting point is used, it fails to detect that two bodies meet.

4.7. Robustness: A Full Simulation Scenario

We set up a scene intended to reflect a typical use-case. The scene consists of a large ground box measuring 4m in each dimension. Four meters above the ground, an XYZ-dragon is launched toward a static XYZ-dragon placed 8m away, with an angular velocity of 50 rad/s and a linear velocity of 16 m/s. We simulate the scene for 300 frames and visualize every frame superimposed onto one image

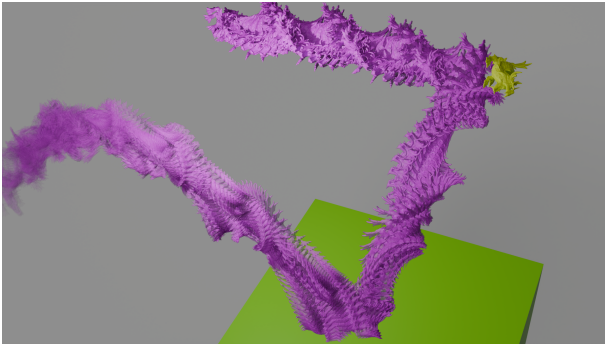


Figure 6: The entire trajectory over 300 simulation frames superimposed into one image.

in Figure 6. The moving dragon initially collides with the static dragon, then bounces and subsequently collides with the ground. Due to the bounce and the high angular velocity, it contacts the ground twice before being launched upwards at approximately 1s.

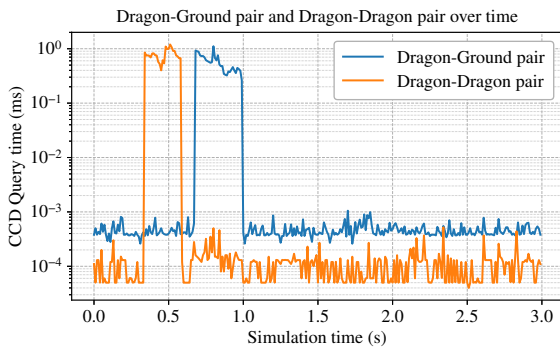


Figure 7: A simulation of a static box-ground SDF and a moving XYZ-dragon SDF shot towards another XYZ-dragon over 300 simulation steps with a time step of 0.01s. Observe how only a few steps near the contacts require around 1ms, whereas remaining steps complete in only a few hundred nanoseconds. This is due to conservative advancement skipping entire simulation steps.

Pairs	Mean > 0.01 ms	Max time (ms)	Mean time (ms)
Dragon-Ground	0.5644 ms	1.1116 ms	0.0606 ms
Dragon-Dragon	0.7828 ms	1.1961 ms	0.0653 ms

Table 8: A simulation of a static box-ground SDF and a moving XYZ-dragon SDF shot towards another XYZ-dragon over 300 simulation steps with a time step of 0.01s. In addition to mean runtime over all simulation frames, we report the subset of frames whose runtime exceeds 0.01ms, since conservative advancement cannot skip the full time step for those frames. We further report the maximum runtime found across all 300 frames. Even for the frames exceeding 0.01 ms, observe how the mean runtime remains slightly below that reported in Table 5 for comparable velocities.

Figure 7 shows that the method spends significant time at contacts. However, it also spends time searching for a time of im-

pact near a contact, but where no contact occurs within the simulated time step. This is expected, since conservative advancement is designed to remain conservative; thus, near potential contacts, it cannot safely advance across an entire time step, and our method must search to determine whether there is a time-of-impact in those steps. Table 8 reports timings for both object pairs in the scene.

4.8. Comparison with PhysX

PhysX is built for real-time performance in applications such as games, where speed is valued over exact geometric accuracy.

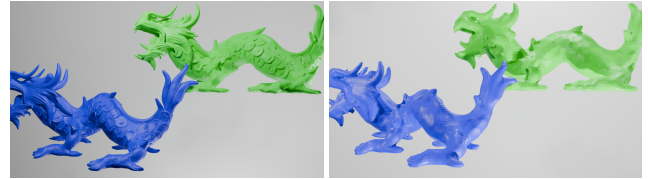


Figure 8: Two moving dragons shot against each other, shown at the time of contact, with a linear velocity of (200, 0, 0) m/s. Observe how they find the exact same correct contact point.

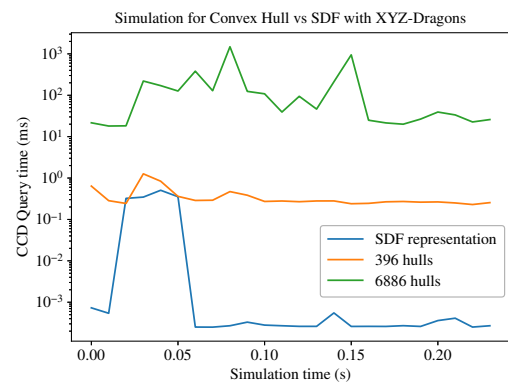


Figure 9: Two moving dragons launched against each-other with a linear velocity of (200, 0, 0) m/s, shown at the time of contact. Observe how the runtime approaches a second for 6886 convex hulls. Despite one contact happening only at 0.04s, PhysX spends a large runtime for the remaining steps, potentially because conservative advancement could be applied to individual convex hulls.

Our method operates under similar constraints targeting fast CCD queries rather than exact solutions, making PhysX a relevant comparison as it reflects typical real-time CCD usage. This allows us to assess whether our method provides advantages under similar performance constraints encountered in real-time applications. Since PhysX does not support CCD for triangle meshes or SDF pairs directly, we approximate each body using convex decomposition. Because PhysX does not expose isolated CCD query timings, reporting them only as part of the broader contact resolution, we control the experimental setup to minimize ambiguity: The two bodies make contact at a single point with only two convex components colliding, as visualized in Figure 8. We report only the CCD query time for our method, excluding contact resolution.

We simulate two identical XYZ-dragons launched toward each

other with a linear velocity of $(200,0,0)$ m/s and zero angular velocity. We evaluate a high-resolution (6,886 hulls) and a low-resolution (396 hulls) decomposition. We note that PhysX may not be optimized for large convex decompositions, and results should be interpreted with this caveat in mind. As shown in Figure 9, our method outperforms even the low-resolution convex decomposition in runtime, and as visualized in Figure 8, the contact is found at the same point. The high-resolution decomposition, while still a poor approximation of the original mesh, incurs runtimes approaching one second rendering it impractical for real-time use.

5. Discussion

As shown by our experiments, our approach is limited by the accuracy of numerically estimated gradients in regular grid-based SDFs as well as the linear convergence rate of gradient descent. Overall, this means our approach can rarely exceed an accuracy of 10^{-5} under limited compute resources. This is acceptable for real-time game-like applications and even permits single-precision arithmetic, yielding substantial memory savings and reduced computing times. Our results also open up questions about improving accuracy: Our optimization approach opens up a range of gradient-type numerical methods and their different tradeoffs, such as rank-one quasi-Newton methods, BFGS, or similar. One may even pursue a Hessian-based approach to get to a full Newton method.

Our experiments further suggest that gradient estimation from discrete SDFs plays an even more significant role. They are cheap to evaluate, and they are sufficiently efficient to get very close to the accurate results. We speculate it may be interesting to have a technique that can use the poor accuracy but fast gradients at a distance, but as one gets closer to a contact, more expensive and higher accuracy techniques can be used. Our method can quickly reduce the number of feasible sample points that can end up as potential contact points. Hence, for the much-reduced number of feasible sample points, a more expensive technique can be afforded without significantly reducing the overall running time.

As shown experimentally, larger grids increase runtime due to worse caching. Our CPU has 20 MB cache, though much of today's hardware offers less, so runtime may worsen for smaller grids on hardware-scarce platforms. Future work should therefore explore alternatives to regular grids, such as octrees, which represent fine details with less memory: Individual lookups may be slower, but fitting more of the SDF in cache could improve overall runtime.

We currently do not provide a GPU implementation. We have attempted to structure the method so that it can be adapted to GPUs as easily as possible. However, the algorithm depends on processing points sequentially in order to maintain the best time-of-impact so far and feed it to other points for early exits, making straightforward parallelization difficult. The benefit of the sequential early exit is that points processed later use an already refined upper bound. In a concurrent execution, all threads begin their iterations before any bound has been established: Once one thread computes an upper bound, other threads have consumed many unnecessary iterations that would have early exited. In practice, the first few starting points require many iterations to converge, but once a valid TOI-upper bound exists, subsequent points terminate in far fewer iterations.

One may improve parallelism by identifying the point most

likely to contact with the SDF, compute its time-of-impact, and then use that estimate as an early exit for the remaining points. This trades some performance, for making points parallelizable. On the other hand, GPUs are well suited for regular grid representations, since 3D textures and hardware-accelerated trilinear interpolation may make point-sampling substantially faster. In this work, we focus on CPU-based implementation because, for games and training simulators, GPUs are often reserved for visual effects. That said, GPU implementations would be interesting to investigate for applications such as physics AI, DRL, and robotics training.

Most other deformable collision detection work based on SDFs uses a local transformation based on mapping model coordinates to spatial coordinates. That mapping emerges naturally from the tetrahedral mesh used for a deformable simulation method. Hence, our approach can use the same technique for mapping our sample points into the undeformed space that stores the SDF in the model coordinates. Alternatively, if fast SDF recomputation is available, that can be used to update the SDFs. That approach is not feasible with our current use of a regular dense grid. A narrow-band or neural representation may be better suited for this purpose. Our technique does introduce the complication that, as surfaces deform, we need to ensure that our point sampling stays on the surface of the model. However, SDFs provide a natural back-projection capability that eliminate any drift from the deforming surface. Even non-optimal back-projection is not a major concern, as our optimization approach would simply trace and move the point sampling on the deformed surface afterwards. As such, our optimization approach may even be more attractive if adaptive SDFs are fast enough. It would entirely eliminate any mapping between local undeformed model coordinates and spatial deformed world coordinates.

Most state-of-the-art rigid body simulators use a first-order semi-implicit time-stepping approach, which lets us treat motion as a screw motion between time endpoints; the assumption exploited in Equation (7). Using higher-order or more implicit methods would require incorporating the effects of forces. This is a current limitation in our work, as with many other simulation frameworks.

6. Conclusion

We have presented a novel method for continuous collision detection between signed distance field pairs, typically achieving runtimes below 1 ms. By formulating CCD as a spatio-temporal optimization problem, our method solves for time of impact using gradient descent over a sparsely sampled collection of starting points on the SDF, together with an effective adaptive temporal step-size. Since no standardized benchmark suite exists for SDF pairs, we evaluated the method using randomized configurations and velocities to assess its robustness across a broad range of cases and to reduce the likelihood of missing important potential edge cases. Our experiments show that sampling only geometric extrema points is sufficient, and in practice, relatively few points are necessary even for geometrically detailed meshes. Across all tested meshes, the method consistently produced time-of-impact estimates close to the true time-of-impact. Overall, the proposed approach robustly handles both simple and complex geometry in real-time on a single CPU core. While our experiments indicate that missed collisions and tunnelling are unlikely, we do not provide a formal guarantee that they cannot occur.

References

- [AEF22] ANDREWS, SHELDON, ERLEBEN, KENNY, and FERGUSON, ZACHARY. “Contact and Friction Simulation for Computer Graphics”. *ACM SIGGRAPH 2022 Courses*. SIGGRAPH ’22. Hybrid Event, Vancouver, Canada: Association for Computing Machinery, 2022. 2.
- [BEB12] BROCHU, TYSON, EDWARDS, ESSEX, and BRIDSON, ROBERT. “Efficient geometrically exact continuous collision detection”. *ACM Trans. Graph.* 31.4 [July 2012]. ISSN: 0730-0301. 2.
- [Bri07] BRIDSON, ROBERT. “Fast Poisson disk sampling in arbitrary dimensions”. *ACM SIGGRAPH 2007 Sketches*. SIGGRAPH ’07. San Diego, California: Association for Computing Machinery, 2007, 22–es. ISBN: 9781450347266. 3.
- [CYN*24] CHEN, XUWEN, YU, CHENG, NI, XINGYU, et al. “A Time-Dependent Inclusion-Based Method for Continuous Collision Detection between Parametric Surfaces”. *ACM Trans. Graph.* 43.6 [Nov. 2024]. ISSN: 0730-0301. 2.
- [FPRJ00] FRISKEN, SARAH F., PERRY, RONALD N., ROCKWOOD, ALYN P., and JONES, THOUIS R. “Adaptively sampled distance fields: a general representation of shape for computer graphics”. *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’00. USA: ACM Press/Addison-Wesley Publishing Co., 2000, 249–254. ISBN: 1581132085. 3.
- [HVTG08] HARMON, DAVID, VOUGA, ETIENNE, TAMSTORF, RASMUS, and GRINSFUND, EITAN. “Robust treatment of simultaneous collisions”. *ACM Trans. Graph.* 27.3 [Aug. 2008], 1–4. ISSN: 0730-0301. 2.
- [KR03] KIM, BYUNGMOON and ROSSIGNAC, JAREK. “Collision prediction for polyhedra under screw motions”. *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*. SM ’03. Seattle, Washington, USA: Association for Computing Machinery, 2003, 4–10. ISBN: 1581137060. 2.
- [LZW*24] LIU, PENGFEI, ZHANG, YUQING, WANG, HE, et al. “Real-time collision detection between general SDFs”. *Computer Aided Geometric Design* 111 [2024], 102305. ISSN: 0167-8396. 3.
- [MEM*20] MACKLIN, MILES, ERLEBEN, KENNY, MÜLLER, MATTHIAS, et al. “Local Optimization for Robust Signed Distance Field Collision”. 3.1 [May 2020]. 2.
- [Mir96] MIRTICH, BRIAN VINCENT. “Impulse-based Dynamic Simulation of Rigid Body Systems”. [1996]. 2, 6.
- [Mus13] MUSETH, KEN. “VDB: High-resolution sparse volumes with dynamic topology”. *ACM Trans. Graph.* 32.3 [July 2013]. ISSN: 0730-0301. 3.
- [Mus21] MUSETH, KEN. “NanoVDB: A GPU-Friendly and Portable VDB Data Structure For Real-Time Rendering And Simulation”. *ACM SIGGRAPH 2021 Talks*. SIGGRAPH ’21. Virtual Event, USA: Association for Computing Machinery, 2021. ISBN: 9781450383738. 3.
- [PFS*19] PARK, JEONG JOON, FLORENCE, PETER, STRAUB, JULIAN, et al. “DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation”. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019. 4.
- [PMA25] PELLETIER-GUÉNÉTTE, JOËL, MERCIER-AUBIN, ALEXANDRE, and ANDREWS, SHELDON. “Real-Time Triangle-SDF Continuous Collision Detection”. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*. SCA ’25. 8.4 [2025]. 3.
- [RKLM04] REDON, STEPHANE, KIM, YOUNG J., LIN, MING C., and MANOCHA, DINESH. “Fast continuous collision detection for articulated models”. *Proceedings of the Ninth ACM Symposium on Solid Modeling and Applications*. SM ’04. Genoa, Italy: Eurographics Association, 2004, 145–156. ISBN: 390567355X. 2.
- [THM23] TRACY, KEVIN, HOWELL, TAYLOR A., and MANCHESTER, ZACHARY. “Differentiable Collision Detection for a Set of Convex Primitives”. *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, 3663–3670. 2.
- [TKM10] TANG, MIN, KIM, YOUNG J., and MANOCHA, DINESH. “Continuous collision detection for non-rigid contact computations using local advancement”. *2010 IEEE International Conference on Robotics and Automation*. 2010, 4016–4021. 2.
- [TMT10] TANG, MIN, MANOCHA, DINESH, and TONG, RUOFENG. “Fast continuous collision detection using deforming non-penetration filters”. *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D ’10. Washington, D.C.: Association for Computing Machinery, 2010, 7–13. ISBN: 9781605589398. 2.
- [WFS*21] WANG, BOLUN, FERGUSON, ZACHARY, SCHNEIDER, TESEO, et al. “A Large-scale Benchmark and an Inclusion-based Algorithm for Continuous Collision Detection”. *ACM Trans. Graph.* 40.5 [Sept. 2021]. ISSN: 0730-0301. 2.
- [WTTM15] WANG, ZHENDONG, TANG, MIN, TONG, RUOFENG, and MANOCHA, DINESH. “TightCCD: Efficient and Robust Continuous Collision Detection using Tight Error Bounds”. *Computer Graphics Forum* 34.7 [2015], 289–298. 2.
- [XB17] XU, HONGYI and BARBIĆ, JERNEJ. “6-DoF Haptic Rendering Using Continuous Collision Detection between Points and Signed Distance Fields”. *IEEE Transactions on Haptics* 10.2 [2017], 151–161. 3, 6.
- [YXYS25] YUAN, XIAODI, XIANG, FANBO, YANG, YIN, and SU, HAO. “C5D: Sequential Continuous Convex Collision Detection Using Cone Casting”. *ACM Trans. Graph.* 44.4 [July 2025]. ISSN: 0730-0301. 2.
- [ZRLK07] ZHANG, XINYU, REDON, STEPHANE, LEE, MINKYOUNG, and KIM, YOUNG J. “Continuous collision detection for articulated models using Taylor models and temporal culling”. *ACM Trans. Graph.* 26.3 [July 2007], 15–es. ISSN: 0730-0301. 2.

7. Supplementary Experiments

7.1. PhysX repeated with angular velocity

We simulate two identical XYZ-dragons launched toward each other with a linear velocity of $(200, 0, 0)$ m/s and an angular velocity of $(200, 0, 0)$ rad/s. We evaluate two convex decompositions of differing fidelity: A high-resolution decomposition (6,886 hulls) and a low-resolution decomposition (396 hulls). We again note that PhysX may not be fully optimized for large convex decompositions, and results should be interpreted with this caveat in mind. As shown in Figure 10, our method is quite similar to low-resolution convex decomposition in terms of runtime close to contact, but is much faster when separated. The high-resolution decomposition, while still a poor visual approximation of the original mesh, incurs runtimes approaching one second, rendering it impractical for real-time use. For both CCD methods, there is not much difference compared to the absence of angular velocity in Figure 9.

7.2. Analytical SDFs

7.2.1. Sphere

As theorized, if our method is sound, the time-of-impact solution for an analytical sphere SDF should be found with little to no error using only two starting points at each pole, for any sphere configuration. Unlike prior experiments, we no longer have to rely on approximate time-of-impact, as finding the point of contact now becomes analytically solving a quadratic equation. Thus instead of only relying on 2000 runs, solving the quadratic equation and using only two starting points, means we can increase this number drastically. We ran the experiment twice: Once without conservative advancement and once with, and we observed no difference in

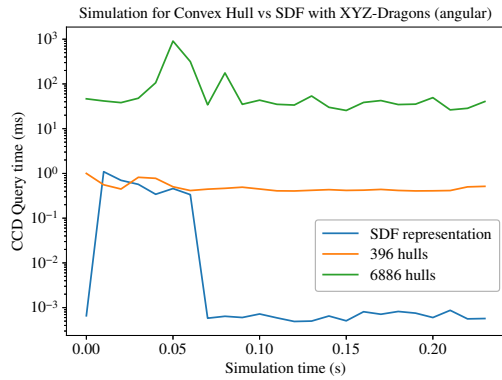


Figure 10: Two moving dragons launched against each other with a linear velocity of $(200, 0, 0)$ m/s and an angular velocity of $(200, 0, 0)$ rad/s, shown at the time of contact. Observe how the runtime approaches a second for 6886 convex hulls. Despite one contact happening only at 0.04s, PhysX spends a large runtime for the remaining steps, potentially because conservative advancement could be applied to individual convex hulls.

results. We report the results without conservative advancement in Table 9.

Points	Min	Max	Mean	Runtime
2	$-1.001 \cdot 10^{-13}$	$2.225 \cdot 10^{-308}$	$-8.313 \cdot 10^{-14}$	0.011 ms

Table 9: Closeness to the ground-truth TOI for 1,000,000,000 random pairs of analytical sphere SDFs, randomly propagated toward another sphere SDF. We measure the differences in mean, min, and max between the analytically computed ground truth time-of-impact and our algorithm's computed time-of-impact. Observe how the min and max clearly show that we do not miss a collision, and we precisely find the time of impact.

Here, we observe many interesting aspects: Firstly, we only use 64-bit floating precision. This means machine epsilon is around $1.11 \cdot 10^{-16}$. This is slightly lower than our mean estimate of 10^{-14} , but it is by no means surprising: Our algorithm performs many floating-point operations, decreasing the amount of precision achievable by quite a lot. This means, the intuition that only two points at the poles are enough for our method to never get stuck is experimentally supported. Out of 1 billion runs, not a single one missed or achieved poor accuracy. We are always within the limits of floating precision error in this benchmark.

7.2.2. Torus

Geometrically, for our algorithm, a torus is a significantly more difficult and interesting test case. Here, both the inner and outer rings are part of the isosurface. Unfortunately, unlike the sphere case, we can not solve the TOI analytically. Thus, we use the same ground truth method as prior experiments, where we take tiny steps. Due to the high accuracy and potentially the tori just grazing, we must take tiny steps of 10^{-12} and refine from that. This means we are only able to run a few thousand configurations for this experiment to be practical.

Points	Min	Max	Mean	Runtime
4	$-1.000 \cdot 10^{-12}$	$6.688 \cdot 10^{-13}$	$-5.152 \cdot 10^{-13}$	0.081 ms

Table 10: Closeness to the ground-truth TOI for 2000 random pairs of analytical torus SDFs, randomly propagated toward another torus SDF. We measure the differences in mean, min, and max between the approximated ground truth time-of-impact and our algorithm's computed time-of-impact. Observe how the min and max clearly show that we do not miss a collision, and we precisely find the time of impact.

Observe how the accuracy is close to the sphere, but slightly worse. This may be limited by the time ground truth computation. Nevertheless, these benchmarks strongly hint that one of the limiting factors in our prior benchmarks are the grid discretization, which decreases the accuracy of the gradient compared to other representations.