Computer-Aided Design 170 (2024) 103685

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Research Paper

Convex Body Collision Detection Using the Signed Distance Function

Pedro López-Adeva Fernández-Layos *, Luis F.S. Merchante

Institute for Research in Technology, Universidad Pontificia de Comillas, Alberto Aguilera 23, 28015, Madrid, Spain

ARTICLE INFO

Keywords:
Signed distance function
Collision detection
Ellipsoid method

ABSTRACT

We present a new algorithm to compute the minimum distance and penetration depth between two convex bodies represented by their Signed Distance Function (SDF). First, we formulate the problem as an optimization problem suitable for arbitrary non-convex bodies, and then we propose the ellipsoid algorithm to solve the problem when the two bodies are convex. Finally, we benchmark the algorithm and compare the results in collision detection against the popular Gilbert–Johnson–Keerthi (GJK) and Minkowski Portal Refinement (MPR) algorithms, which represent bodies using the support function. Results show that our algorithm has similar performance to both, providing penetration depth like MPR and, with better robustness, minimum distance like GJK. Our algorithm provides accurate and fast collision detection between implicitly modeled convex rigid bodies and is able to substitute existing algorithms in previous applications whenever the support function is replaced with the SDF.

1. Introduction

Collision detection is a fundamental problem whenever non-penetration constraints are imposed in physics simulations. In applications demanding real-time or even faster simulation, like video games, virtual reality, or robotics, speed is never enough, and collisions are usually restricted to convex bodies. If geometry is non-convex then it is approximated or decomposed into convex parts [1].

The most popular algorithms for convex body collision detection are Minkowski Portal Refinement (MPR) [2] and GJK [3] and independent C/C++ implementations can be found in LibCCD [4] (GJK/MPR), Bullet [5] (GJK), PhysX [6] (GJK), JoltPhysics [7] (GJK) and SimBody [8] (MPR). The only inputs that both MPR and GJK require to operate are the bodies support functions. The support function can be efficiently defined for a variety of primitives and transformations and constitutes a complete representation of the convex body geometry. This flexibility in geometry representation could explain why they have become more popular than the similarly fast Lin-Canny [9] and V-Clip [10], which are restricted to polyhedra.

The support function is also defined for non-convex bodies but it only depends on the convex hull of the body and collision detection using the support function will return collisions between the convex hulls, which may not be sufficient in all situations. The SDF is an alternative representation, powerful enough to represent non-convex geometry, with many interesting properties not only related to collision detection [11]. It is by definition the solution to the body/particle distance queries. Other types of collisions also have exact algorithms like ray/SDF intersections [12], for ray tracing, or segment/SDF [13] for particle continuous collision detection, but it is not obvious how to extend it to body/body problems.

Previous work for body/body problems using the SDF has centered on the general non-convex case. The most basic approach is sampling points from one body and testing against the SDF of the other one, which may not detect all collisions [14]. When one of the bodies is represented by a mesh [15] also considers edge/SDF intersections and [16] also considers face/SDF. Both approaches give approximate results which improve as the mesh is refined.

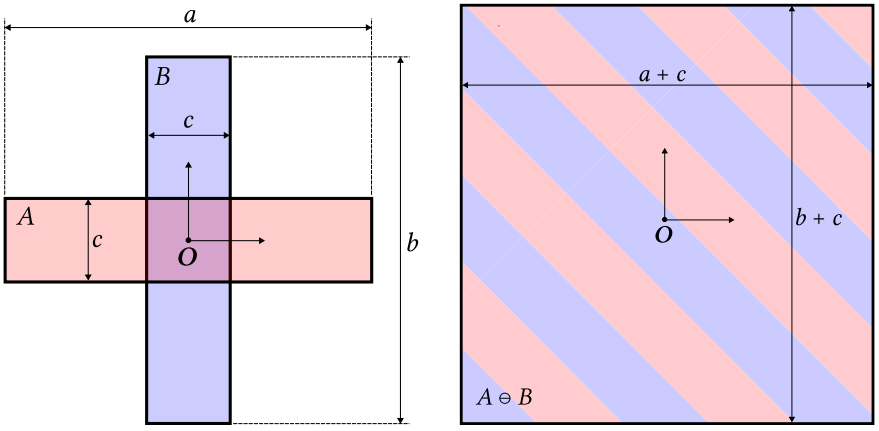
The main objective of this article is to offer a fast, exact, and robust algorithm for collision detection between continuous bodies using only the SDF representation. We will sacrifice generality however and restrict ourselves to convex bodies. This loses the main advantage of the SDF over the support function but, in our opinion, it is a necessary first step towards the much more difficult general non-convex problem. We also give an optimization-based formulation that constitutes a good starting point towards that goal.

2. Background

Given vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ the scalar product is $\boldsymbol{u} \cdot \boldsymbol{v}$ and the norm $\|\boldsymbol{u}\| = \sqrt{\boldsymbol{u} \cdot \boldsymbol{u}}$. The origin point is $\boldsymbol{O}$ and other points $\boldsymbol{O} + \boldsymbol{u}$ are written simply using the vector $\boldsymbol{u}$. Given a set of points A the boundary is ∂A, the interior A° and the complement A^{c}. We define $-A = \{-\boldsymbol{x} : \boldsymbol{x} \in A\}$ and $A + \boldsymbol{u} = \{\boldsymbol{x} + \boldsymbol{u} : \boldsymbol{x} \in A\}$. The closed ball with center $\boldsymbol{c}$ and radius

* Corresponding author.
E-mail addresses: plopezadeva@comillas.edu (P. López-Adeva Fernández-Layos), lfsanchez@comillas.edu (L.F.S. Merchante).

https://doi.org/10.1016/j.cad.2024.103685
Received 6 November 2023; Received in revised form 14 January 2024; Accepted 30 January 2024
Available online 2 February 2024



(a) Two rectangles of equal width c but different lengths a and b interlock at their centers. Any small translation will not alter the intersection (a square with a side length c) and the minimum translation that solves the penetration is vertical if $a > b$ or horizontal if $a < b$.

(b) The penetration depth is given by the minimum distance of $\boldsymbol{O}$ to the boundary of $A \ominus B$.

Fig. 1. Translational interpenetration depth. Even in the convex case it depends on the global shape of the bodies and not just their intersection (a). The problem can be formulated using the Minkowski difference of the bodies (b).

R is $S(\boldsymbol{c}, R) = \{\boldsymbol{x} : \|\boldsymbol{x} - \boldsymbol{c}\| \leq R\}$ and the closed halfspace with normal $\boldsymbol{h}$ at a distance δ from $\boldsymbol{O}$ is written $\mathcal{H}(\boldsymbol{h}, \delta) = \{\boldsymbol{x} : \boldsymbol{h} \cdot \boldsymbol{x} \leq \delta\}$

2.1. Collision detection

We make a brief review of the collision detection problem. Comprehensive practical overviews can be found in [17,18], which also describe in detail the GJK algorithm. [19] reviews the problem for wider applications and more focus on complexity results. Because of their popularity and for concreteness, we will focus on the GJK and MPR algorithms to extract the desired qualities in a real-time collision detection algorithm.

2.1.1. Problem description

Two closed bodies A and B are colliding if $A \cap B \neq \emptyset$, with penetration if $A^o \cap B^o \neq \emptyset$ or only touching if $A^o \cap B^o = \emptyset$. If there is no interpenetration we usually want a pair of closest points between them. If the bodies have collided we want some measure of interpenetration. Penetration usually constitutes an error to be corrected and no objective measure of penetration exists but a common definition is translational penetration: the minimum linear displacement necessary to correct interpenetration [20].

In general, it is more difficult to solve the penetration problem as it depends on the global shape of both bodies and not just on their intersection. This is true for deep penetrations even for simple shapes, as shown in Fig. 1(a).

We can answer all the queries with the following interface: given two bodies A and B as inputs we output a flag indicating if they are in collision and two points $\boldsymbol{a} \in A, \boldsymbol{b} \in B$ where $\boldsymbol{a} - \boldsymbol{b}$ is the minimum norm vector such that $B + (\boldsymbol{a} - \boldsymbol{b})$ touches A.

2.2. Configuration space obstacle

Given two sets of points A and B their Minkowski addition is $A \oplus B = \{\boldsymbol{x} + \boldsymbol{y} : \boldsymbol{x} \in A, \boldsymbol{y} \in B\}$ which can also be understood as a convolution between the two sets since $A \oplus B = \bigcup_{\boldsymbol{x} \in A} (B + \boldsymbol{x}) = \bigcup_{\boldsymbol{x} \in B} (A + \boldsymbol{x})$. In general if $A = \bigcup_\alpha A_\alpha$ and $B = \bigcup_\beta B_\beta$ then $A \oplus B = \bigcup_{\alpha,\beta} A_\alpha \oplus B_\beta$. If A and B are convex then $A \oplus B$ is convex and if we displace B by an amount $\Delta\boldsymbol{x}$ then the Minkowski addition gets displaced the same amount: $A \oplus (B + \Delta\boldsymbol{x}) = (A \oplus B) + \Delta\boldsymbol{x}$.

The Minkowski difference, also known as the (Translational) Configuration Space Obstacle (CSO), is $A \ominus B = A \oplus (-B) = \{\boldsymbol{x} - \boldsymbol{y} : \boldsymbol{x} \in A, \boldsymbol{y} \in B\}$. We can reformulate the collision problem in terms of $C = A \ominus B$: the distance of $\boldsymbol{O}$ to ∂C gives the penetration depth when $\boldsymbol{O} \in C$ and the separation distance when $\boldsymbol{O} \notin C$. A problem between two bodies is reduced to a problem between a point and a body.

Computing explicitly the Minkowski difference is usually complicated but for two balls we have that

$$S(\boldsymbol{c}_a, R_a) \ominus S(\boldsymbol{c}_b, R_b) = S(\boldsymbol{c}_a - \boldsymbol{c}_b, R_a + R_b) \tag{1}$$

2.2.1. GJK

The original GJK algorithm answers the collision and minimum distance exactly [3] and an enhanced version gives the translational penetration approximately [21] or exactly [22].

It requires the support function of both bodies. For some body $\Omega \subset \mathbb{R}^n$ the support function s maps any vector in $\mathbb{R}^n$ to a scalar:

$$s(\boldsymbol{v}) = \max_{\boldsymbol{x} \in \Omega} \boldsymbol{x} \cdot \boldsymbol{v} \tag{2}$$

We are usually interested in the support points $\sigma(\boldsymbol{v})$, the maximizers of the support function:

$$\sigma(\boldsymbol{v}) = \operatorname*{argmax}_{\boldsymbol{x} \in \Omega} \boldsymbol{x} \cdot \boldsymbol{v} \tag{3}$$

Frequently σ is multivalued, but any value is a valid support point. The support function can be efficiently evaluated for a variety of bodies and in particular polytopes.

The state of the GJK algorithm is a simplex of at most $n + 1$ points (a tetrahedron in 3D) contained inside the CSO of the two bodies. The simplex is moved towards the origin until it either contains it, a collision, or it is proved that the origin lies outside. It requires a finite number of steps if the bodies are polyhedra or an error tolerance otherwise.

Simulations usually proceed in small time steps during which most objects make small displacements (temporal coherence). GJK has been extended to exploit this feature [21] and can run incremental queries in almost constant time. Practical considerations like this make the algorithm more attractive than alternatives with optimal complexity [23].

Robustness is hard to achieve in geometrical problems. Numerical errors due to finite precision can invalidate assumptions like convexity, orthonormality, or affine independence. Robustness issues with GJK are

well known and [24] gives a comprehensive review of previous work on the problem and modifies GJK to improve the robustness.

2.2.2. MPR

The MPR algorithm [2] has a lot of similarities with GJK but sacrifices some functionality, computing the distance between non-colliding bodies, in exchange for better robustness and implementation simplicity. It also operates with the support function and also maintains at each step a simplex which is evolved in a simpler way than the GJK algorithm. As the GJK algorithm, it finishes in a finite number of steps if the bodies are polyhedra and can run incremental queries. Although it does not return the minimum distance between non-colliding bodies it can return in case of collision, with some additional iterations, the minimum translation along a given direction that puts the bodies in a contact state. Taking the direction that passes between two deep interior points of the bodies usually gives a good approximation of the minimum translational depth.

2.3. The signed distance function

Given some closed body $\Omega \subset \mathbb{R}^n$ with surface $\partial\Omega$ its SDF ϕ is a scalar function defined for any point $\boldsymbol{x} \in \mathbb{R}^n$ as:

$$\phi(\boldsymbol{x}) = \begin{cases} +\min\limits_{\boldsymbol{y}\in\partial\Omega} |\boldsymbol{x}-\boldsymbol{y}| & \text{if } \boldsymbol{x} \notin \Omega \\ -\min\limits_{\boldsymbol{y}\in\partial\Omega} |\boldsymbol{x}-\boldsymbol{y}| & \text{if } \boldsymbol{x} \in \Omega \end{cases} \tag{4}$$

The SDF gives a complete implicit representation of the geometry since $\Omega = \{\boldsymbol{x} : \phi(\boldsymbol{x}) \leq 0\}$ and $\partial\Omega = \{\boldsymbol{x} : \phi(\boldsymbol{x}) = 0\}$.

The projection of point $\boldsymbol{x}$ onto $\partial\Omega$ is

$$p(\boldsymbol{x}) = \arg\min_{\boldsymbol{y}\in\partial\Omega} |\boldsymbol{x}-\boldsymbol{y}| \tag{5}$$

The set of points where $p(\boldsymbol{x})$ is not unique is known as the cut locus.

The SDF is 1-Lipschitz continuous and therefore almost everywhere differentiable (Rademacher's theorem). If $\boldsymbol{x} \in \partial\Omega$ then $\nabla\phi(\boldsymbol{x})$ is equal to the outward normal of the surface. For $\boldsymbol{x} \notin \partial\Omega$ if $p(\boldsymbol{x})$ is unique then $\phi(\boldsymbol{x})$ is at least C^1 and when $\partial\Omega$ is C^k smooth the SDF is also C^k smooth [25]. When the gradient is defined we have that $|\nabla\phi| = 1$ (Eikonal equation) which allows us to compute the projection as $p(\boldsymbol{x}) = \boldsymbol{x} - \phi(\boldsymbol{x})\nabla\phi(\boldsymbol{x})$. If $\Omega \subseteq \Omega'$ then $\phi_\Omega(\boldsymbol{x}) \leq \phi_{\Omega'}(\boldsymbol{x})$ for all $\boldsymbol{x}$.

When Ω is convex we have additional properties [26]: the projection of any $\boldsymbol{x} \notin \Omega$ is unique and therefore the gradient is always defined on the outside. Inside Ω the set of points with non-unique projection is known as the skeleton of Ω [27], which must be non-empty unless Ω is a halfspace (Fig. 2). Even when the gradient is undefined it is easy to find a subgradient $\partial\phi$: for $\boldsymbol{x} \in \partial\Omega$ any normal of a supporting halfspace and for $\boldsymbol{x} \notin \partial\Omega$ the vector $(\boldsymbol{x} - p(\boldsymbol{x}))/\phi(\boldsymbol{x})$ where $p(\boldsymbol{x})$ can be any projection of $\boldsymbol{x}$ if not unique. Any convex combination of subgradients is also subgradient and we have $|\partial\phi| \leq 1$. Projection onto convex sets is non-expansive:

$$\boldsymbol{x}_1, \boldsymbol{x}_2 \notin \Omega \implies \|p(\boldsymbol{x}_1) - p(\boldsymbol{x}_2)\| \leq \|\boldsymbol{x}_1 - \boldsymbol{x}_2\| \tag{6}$$

Finally, Ω is convex if and only if ϕ is convex.

The main takeaway of this section is that the SDF has properties attractive for optimization since the gradient is almost everywhere defined. Additionally when Ω is convex the SDF is also convex and it is very easy to find a subgradient on the small chance that the gradient is undefined.

3. Ellipsoid method

We briefly overview the ellipsoid method for convex optimization, which works by obtaining a sequence of shrinking ellipsoids containing at all times the optimum (Fig. 3(a)). An in-depth exposition can be found in [28] and for convex optimization in general we refer the reader to [29].

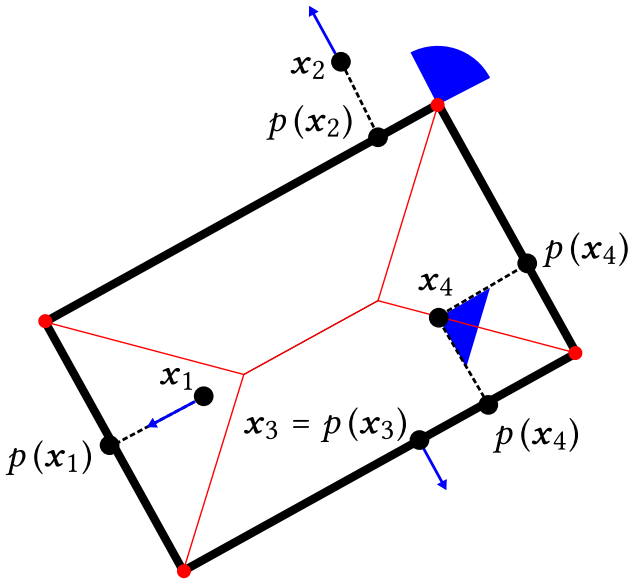


Fig. 2. In red the points where the gradient is undefined: the skeleton and the 4 corners on the surface. $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ have unique projections and the gradient is defined. The gradient at $\boldsymbol{x}_3$ is the outward normal. $\boldsymbol{x}_4$ has two projections and no gradient. In blue the unique gradient or the convex set of subgradients. The gradient is defined almost everywhere and the subgradient is easily computed choosing any projection. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.1. Ellipsoid representation and basic operations

We will consider an ellipsoid $E \subset \mathbb{R}^n$ as the transformation through an affine map of the unit ball.

$$E = \{\boldsymbol{M}\boldsymbol{x} + \boldsymbol{e} : \|\boldsymbol{x}\| \leq 1\} \tag{7}$$

Where the parameters are $\boldsymbol{M}$, an $n \times n$ matrix, and $\boldsymbol{e}$ a vector of size n. Making the change $\boldsymbol{x} \to \boldsymbol{M}^{-1}(\boldsymbol{x} - \boldsymbol{e})$ and calling $\boldsymbol{P} = \boldsymbol{M}\boldsymbol{M}^T$ we get:

$$\mathcal{E}(\boldsymbol{e}, \boldsymbol{P}) = \left\{\boldsymbol{x} : (\boldsymbol{x} - \boldsymbol{e})^T \boldsymbol{P}^{-1} (\boldsymbol{x} - \boldsymbol{e}) \leq 1\right\} \tag{8}$$

We can compute the minimum bounding ellipsoid $\mathcal{E}(\boldsymbol{e}', \boldsymbol{P}')$ of $\mathcal{E}(\boldsymbol{e}, \boldsymbol{P}) \cap \mathcal{H}(\boldsymbol{h}, \delta)$ using Algorithm 1 (see Fig. 4).

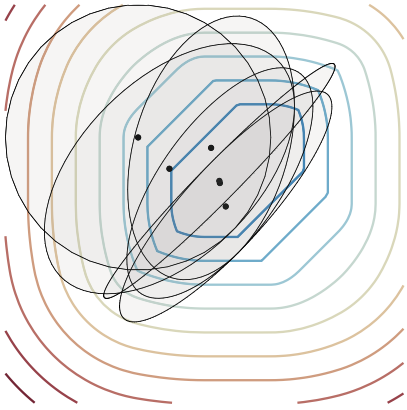
Algorithm 1 Minimum bounding ellipsoid for the intersection of an ellipsoid and a half-plane

procedure CUT($\boldsymbol{e}, \boldsymbol{P}, \delta, \boldsymbol{h}$)
$\alpha \leftarrow \frac{\boldsymbol{e}\cdot\boldsymbol{h} - \delta}{\sqrt{\boldsymbol{h}^T \boldsymbol{P} \boldsymbol{h}}}$
$\boldsymbol{g} \leftarrow \frac{\boldsymbol{P}\boldsymbol{h}}{\sqrt{\boldsymbol{h}^T \boldsymbol{P} \boldsymbol{h}}}$
$\boldsymbol{e}' \leftarrow \boldsymbol{e} - \frac{1+n\alpha}{n+1}\boldsymbol{g}$
$\boldsymbol{P}' \leftarrow \frac{n^2}{n^2-1}(1-\alpha^2)\left(\boldsymbol{P} - \frac{2(1+n\alpha)}{(n+1)(1+\alpha)}\boldsymbol{g}\boldsymbol{g}^T\right)$
return $\boldsymbol{e}', \boldsymbol{P}'$
end procedure

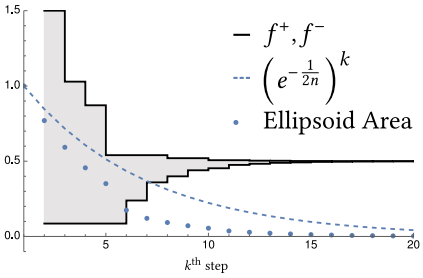
We say we have shallow cuts for $-1/n < \alpha < 0$, central cuts when $\alpha = 0$, and deep cuts for $\alpha > 0$. It is usually convenient to apply the deepest cut possible as it gets smaller ellipsoids. A fundamental fact is that the new ellipsoid has strictly less volume. When $\alpha \geq 0$ we have that:

$$\frac{\text{vol}(E')}{\text{vol}(E)} < e^{-\frac{1}{2n}} < 1 \tag{9}$$

We will see that we can always make at least central cuts and so Eq. (9) ensures convergence. Notice that convergence speed decreases exponentially with the dimensionality n which explains why the ellipsoid method is usually of limited practical utility. For our purposes, however, we just have $n = 2$ or $n = 3$.



(a) The ellipsoid method works by generating a sequence of smaller in area ellipses that always contain the minimum.



(b) The error $f^+ - f^-$ decreases exponentially, even though it can remain constant during some steps. The ellipsoid area decreases strictly.

Fig. 3. Example of the ellipsoid method in two dimensions.

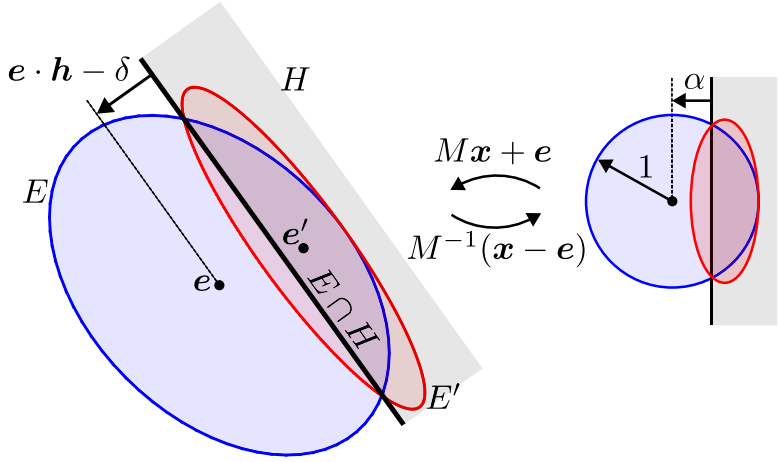


Fig. 4. Cut of ellipsoid E with halfspace H. The new ellipsoid E', computed using Algorithm 1, is the smallest one that contains $E \cap H$. Parameter α is the cut depth relative to the unit ball that transforms to E through an affine function.

The minimum of a linear function $\boldsymbol{c}\cdot\boldsymbol{x}$ over the ellipsoid is easily determined by:

$$\min\{\boldsymbol{c}\cdot\boldsymbol{x} : \boldsymbol{x}\in E\} = \boldsymbol{c}\cdot\boldsymbol{e} - \sqrt{\boldsymbol{c}^T\boldsymbol{P}\boldsymbol{c}} \tag{10}$$

3.2. Algorithm

We state our (unconstrained) convex optimization problem as:

$$\boldsymbol{x}^\star = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{11}$$

The optimal value is $f^\star = f(\boldsymbol{x}^\star)$. We will call f^+ an upper bound and f^- a lower bound estimate:

$$f^+ \geq f^\star \geq f^- \tag{12}$$

Initially, we set $f^+ \leftarrow \infty$ and $f^- \leftarrow -\infty$. We start with an ellipsoid E such that $\boldsymbol{x}^\star \in E$.

If we evaluate f and ∂f at $\boldsymbol{e}$ we get a maybe better upper bound $f^+ \leftarrow \min(f^+, f(\boldsymbol{e}))$ and a first-order approximation h, which by convexity is a lower bound of f for all $\boldsymbol{x}$:

$$h(\boldsymbol{x}) = f(\boldsymbol{e}) + \partial f(\boldsymbol{e})\cdot(\boldsymbol{x}-\boldsymbol{e}) \leq f(\boldsymbol{x}) \tag{13}$$

Since $h(\boldsymbol{x}) \leq f(\boldsymbol{x})$ the minimum of h over E is a lower bound of $f^\star$. Using Eq. (10):

$$f^- \leftarrow \max\left(f^-, f(\boldsymbol{e}) - \sqrt{\partial f(\boldsymbol{e})^T\boldsymbol{P}\partial f(\boldsymbol{e})}\right) \tag{14}$$

Define the halfspace H as

$$H = \left\{\boldsymbol{x} : h(\boldsymbol{x}) \leq f^+\right\} = \mathcal{H}\left(\partial f(\boldsymbol{e}), f^+ - f(\boldsymbol{e}) + \partial f(\boldsymbol{e})\cdot\boldsymbol{e}\right) \tag{15}$$

Then from Eq. (13) we see that $\boldsymbol{x}\notin H \implies f(\boldsymbol{x}) > f^+$ and equivalently $\boldsymbol{x}^\star \in H$. Our new ellipsoid E' will be the minimum volume one that contains $E\cap H$, which is given by $(\boldsymbol{e}', \boldsymbol{P}') = \text{cut}\left(\boldsymbol{e}, \boldsymbol{P}, f^+ - f(\boldsymbol{e}) + \partial f(\boldsymbol{e})\cdot\boldsymbol{e}, \partial f(\boldsymbol{e})\right)$.

The ellipsoid algorithm pseudo-code is given in Algorithm 2. The stopping criterion is that the maximum error $f^+ - f^-$ must be below some user-defined tolerance Δ. A graphical representation is given in Fig. 5.

4. Our approach using the SDF

We now discuss three possible approaches to solving collision problems using the SDF: evaluating the support function with the SDF to use existing algorithms, formulating a constrained optimization problem, and finally solving an unconstrained optimization problem.

4.1. Computing the support function using the SDF

One possible approach is using the SDF to compute support points and reuse any of the well-tested existing algorithms. We could simply

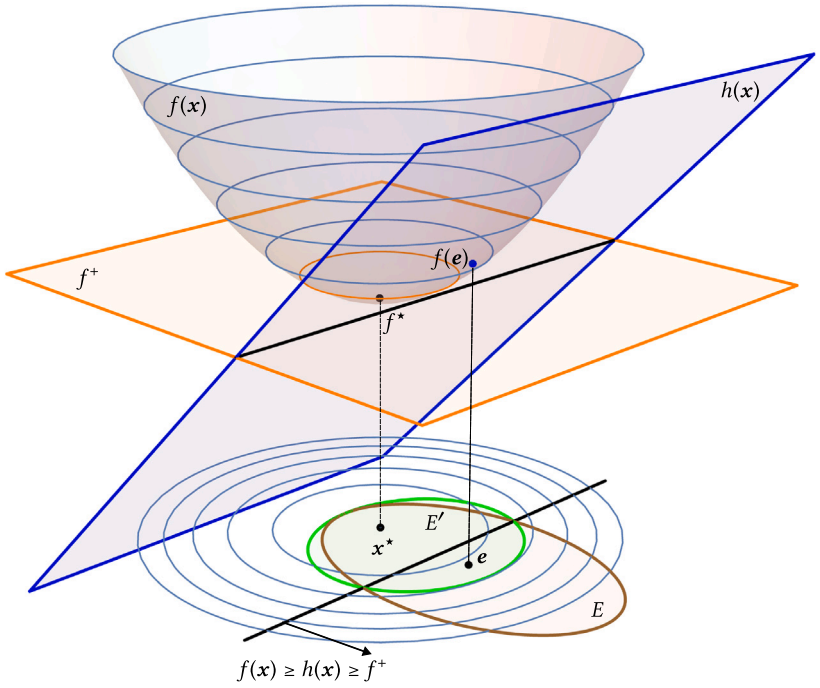


Fig. 5. Ellipsoid algorithm step: we start with an ellipse E, in brown, and an upper bound f^+, in orange. We evaluate f on the center e of E. The tangent plane $h(x)$, in blue, is a lower bound of $f(x)$. We can safely discard from our search all points where $h(x) \geq f^+$ to obtain a new ellipsoid E', in green. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Algorithm 2 Ellipsoid method

procedure MINIMIZE(f, e^0, P^0, Δ)
  $e, P \leftarrow e^0, P^0$ ▷ Initialize ellipsoid
  $f^-, f^+ \leftarrow -\infty, +\infty$ ▷ Initialize optimum bounds
  while $f^+ - f^- > \Delta$ **do**
    $f^+ \leftarrow \min\left(f^+, f(e)\right)$
    $f^- \leftarrow \max\left(f^-, f(e) - \sqrt{\partial f(e)^T P \partial f(e)}\right)$
    $e, P \leftarrow \text{cut}\left(e, P, f^+ - f(e) + \partial f(e) \cdot e, \partial f(e)\right)$
  end while
  return f^-, f^+
end procedure

solve the problem:

$$\sigma(v) = \underset{\phi(x)\leq 0}{\text{argmax}}\ x \cdot v \tag{16}$$

Solving an optimization problem for each support function evaluation would be too expensive. In Appendix A we show that it is possible to compute fast approximations of the support function as a limit of the SDF, but unfortunately, if the limit is evaluated numerically, the errors will affect the precision of collision detection. Additionally, this approach cannot be extended for non-convex collisions in the future. We find the result interesting but we have discarded it in favor of a more direct approach.

4.2. Constrained problem

Given two closed bodies A and B with SDFs ϕ_A and ϕ_B there are multiple ways to formulate the different distance queries. Just collision detection is naturally modeled as a feasibility problem:

$$\begin{aligned} &\text{find} && x \\ &\text{subject to} && \phi_A(x) \leq 0 \\ & && \phi_B(x) \leq 0 \end{aligned} \tag{17}$$

A natural approach to minimum distance is:

$$\begin{aligned} &\text{minimize} && \|x - y\| \\ &\text{subject to} && \phi_A(x) \leq 0 \\ & && \phi_B(y) \leq 0 \end{aligned} \tag{18}$$

Problem (18) however doubles the dimensionality relative to Problem (17) and still does not give any penetration information when the bodies collide. It is interesting nevertheless, as it will return exact distances between the bodies for any implicit functions ϕ_A and ϕ_B, not just SDFs.

We now give a formulation for minimum distance that also return penetration depth by defining a new function ϕ_{AB}.

4.3. Minimum distance and penetration depth

The function $\phi_{AB}(x) = \max\left(\phi_A(x), \phi_B(x)\right)$ is an implicit equation for $A \cap B$ [30]. It is not the unique one [31] but if $\phi_{A\cap B}(x)$ is the SDF of $A \cap B$ then we have that [12] $\phi_{AB}(x) \leq \phi_{A\cap B}(x)$ for all x and with equality when $x \in A \cap B$. We have the following result:

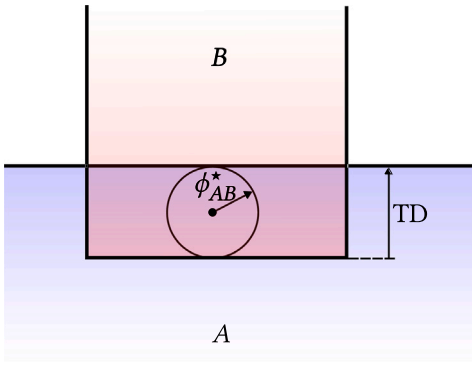
Proposition 1. *If $\phi^\star_{AB} \leq 0$ there is a collision and $S\left(x^\star, -\phi^\star_{AB}\right)$ is the biggest inscribed sphere in $A \cap B$. If $\phi^\star_{AB} > 0$ it gives half the minimum distance between A and B (proof in Appendix B.1).*

The maximum radius inscribed sphere of $A \cap B$ (Chebyshev sphere) gives a lower bound approximation of the intersection volume which can be used as a measure of penetration for collision resolution [32] [33]. It is always an underestimator of the minimum translational depth but it agrees on some penetration instances like shown in Fig. 6.

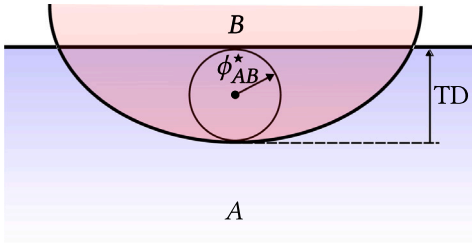
4.3.1. Initial search region with bounding spheres

We require bounding spheres for the collision bodies A and B. They may not be tight nor have the center inside the bodies. If $A \subseteq S_A = S\left(c_A, R_A\right)$ and $B \subseteq S_B = S\left(c_B, R_B\right)$ we can use the SDFs of the spheres as lower bounds $\underline{\phi}_A$ and $\underline{\phi}_B$ of the bodies SDFs:

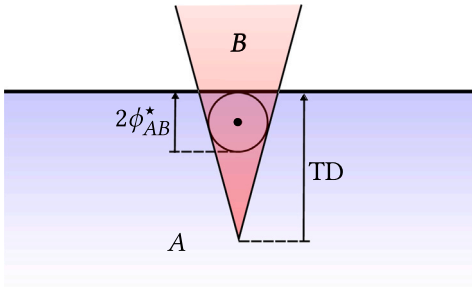
$$\underline{\phi}_A(x) = \|x - c_A\| - R_A \leq \phi_A(x)$$
$$\underline{\phi}_B(x) = \|x - c_B\| - R_B \leq \phi_B(x)$$



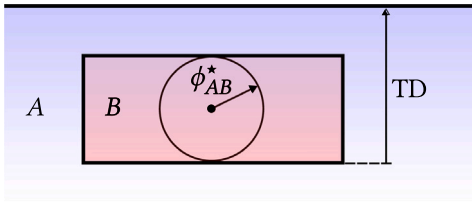
(a) For small flat penetrations $2\phi^\star_{AB} = \text{TD}$



(b) Small penetration between smooth low curvature objects also makes both penetration definitions agree.



(c) Penetration of sharp objects makes both definitions disagree and we have $2\phi^\star_{AB} < \text{TD}$.



(d) With deep penetration of objects $2\phi^\star_{AB} < \text{TD}$.

Fig. 6. Maximum radius inscribed sphere ($R = \phi^\star_{AB}$) comparison against minimum translational depth (TD) on different penetration instances.

The furthest point between the spheres gives an upper bound value $\phi^+ = \|c_A - c_B\| + R_A + R_B \geq \phi^\star$, which is extremely pessimistic. If the bodies are convex and the balls are near enough optimal then $c_A \in A$ and $c_B \in B$ and $\phi^+ = \|c_A - c_B\|$. We usually know even better bounds from previous collision states.

The following results related to the minimum bounding ellipsoid of the intersection of two spheres give valid initial ellipsoid for our optimization algorithm.

Algorithm 3 Minimum bounding ellipsoid for the intersection of two spheres.

procedure EllipsoidBound(c_A, c_B, R_A, R_B)
  if $R_B > R_A$ **then** ▷ Swap spheres to ensure $R_B \leq R_A$
    $c_A, c_B, R_A, R_B \leftarrow c_B, c_A, R_B, R_A$
  end if
  $d \leftarrow \|c_A - c_B\|$
  if $d > R_A + R_B$ **then** ▷ Empty intersection
    return error
  end if
  $s \leftarrow \frac{R_A^2 - R_B^2}{d}$
  $d_A \leftarrow \min\left(d, \frac{d+s}{2}\right)$
  $d_B \leftarrow \max\left(0, \frac{d-s}{2}\right)$
  $u \leftarrow \frac{c_B - c_A}{d}$
  $e \leftarrow c_A + d_A u$
  $P \leftarrow (R_B^2 - d_B^2)\left[I - \frac{2d_B}{R_B + d_B} uu^T\right]$
  return e, P
end procedure

Proposition 2. *Algorithm 3 computes the minimum bounding ellipsoid of the intersection of two spheres (proof in Appendix B.2).*

Proposition 3. $x^\star_{AB} \in \mathit{EllipsoidBound}\left(c_A, c_B, R_A + \phi^+_{AB}, R_B + \phi^+_{AB}\right)$ *(proof in Appendix B.3).*

4.3.2. Initial search region with bounding halfspaces

We now assume we know a bounding halfspace for A, another for B, and some upper bound ϕ^+. We state the proposition without proof, as it is completely analogous to the same propositions with bounding spheres, when $A \subseteq \mathcal{H}\left(h_A, \delta_A\right)$ and $B \subseteq \mathcal{H}\left(h_B, \delta_B\right)$.

Proposition 4. $x^\star_{AB} \in \mathcal{H}\left(h_A, \delta_A + \phi^+_{AB}\right) \cap \mathcal{H}\left(h_B, \delta_B + \phi^+_{AB}\right)$

4.4. Incremental collisions

In this section we study what happens when body B is displaced by a small amount Δx to obtain body $B' = B + \Delta x$ with SDF $\phi_{B'}(x) = \phi_B(x - \Delta x)$. The function $\phi_{AB'}$ will achieve a new minimum $\phi^\star_{AB'} = \phi^\star_{AB} + \Delta\phi^\star_{AB}$ at some point $x^\star_{AB} + \Delta x^\star_{AB}$. The following results give a new initial upper bound after the displacement.

Proposition 5. $\left|\Delta\phi^\star_{AB}\right| \leq \|\Delta x\|/2$ *(proof in Appendix B.4).*

Corollary 1. *After a displacement Δx a starting upper bound of the new problem is*

$$\phi^+_{AB'} = \frac{\phi_A(x^\star_{AB}) + \phi_B(x^\star_{AB}) + \|\Delta x\|}{2} \tag{19}$$

(proof in Appendix B.4)

4.5. Approximating the SDF

One useful property of ϕ_{AB} is that it is an R-function [31], meaning that the sign of ϕ_{AB} depends only on the sign of ϕ_A and ϕ_B. In

particular $\phi_{AB} < 0$ if and only if $\phi_A < 0$ and $\phi_B < 0$. This relaxes the requirement of using true SDFs since the collision state can be equally determined with any implicit function. Of course, the distance or penetration result will be as accurate as our implicit function approximates the true SDF.

For a convex body described by an (infinite, uncountable) set of halfspaces:

$$A = \bigcap_{\alpha} \mathcal{H}\left(\boldsymbol{h}_\alpha, \delta_\alpha\right) \tag{20}$$

The following implicit function is well-known and satisfies $\tilde{\phi}_A(\boldsymbol{x}) = \phi_A(\boldsymbol{x})$ for $\boldsymbol{x} \in A$ and $\tilde{\phi}_A(\boldsymbol{x}) < \phi_A(\boldsymbol{x})$ if $\boldsymbol{x} \notin A$:

$$\tilde{\phi}_A(\boldsymbol{x}) = \max_{\alpha} \boldsymbol{h}_\alpha \cdot \boldsymbol{x} - \delta_\alpha \tag{21}$$

Note that any half-space of the collection gives a lower bound approximation of the SDF. We use the following approximation for polyhedra: assuming the origin is inside the polyhedron consider the ray starting at the origin passing through point $\boldsymbol{x}$ then the intersection distance of the ray with plane α is

$$t(\alpha) = \frac{\delta_\alpha}{\boldsymbol{h}_\alpha \cdot \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}}$$

Let $\alpha^\star$ be the minimizer of the problem

$$\begin{aligned} &\text{minimize} && t(\alpha) \\ &\text{subject to} && t(\alpha) \geq 0 \end{aligned} \tag{22}$$

Then we use the approximation $\tilde{\phi}_A(\boldsymbol{x}) = \boldsymbol{h}_{\alpha^\star} \cdot \boldsymbol{x} - \delta_{\alpha^\star}$. It has the same sign as the true SDF and we can use Algorithm 4 that uses the connectivity information between the faces and the previous query results.

Algorithm 4 Polyhedron approximate SDF

```
procedure PolyhedronSDF(x, h, δ, N, α_0 = 0)
    if α_0 = 0 then                          ▷ Not set, initialize to 1
        α ← 1
    else
        α ← α_0
    end if
    repeat                                   ▷ Maximize s
        s ← h_α · x / δ_α
        for β ∈ N_α do
            s' ← h_β · x / δ_β
            if s' > s then
                s ← s'
                α ← β
            end if
        end for
    until s > 1
    return h_α · x − δ_α                     ▷ Use plane α for SDF value
end procedure
```

Proposition 6. *Given a point $\boldsymbol{x}$ and a convex polyhedron Algorithm 4 evaluates a lower bound of the true SDF and both functions have the same sign at $\boldsymbol{x}$. For each supporting halfspace α of the polyhedron we require:*

- *The halfspace equation is given by $\boldsymbol{h}_\alpha$, δ_α*
- *The collections of neighboring faces N_α*

(proof in Appendix B.5)

5. Benchmarks

In this section we compare our SDF based algorithm against GJK and MPR. Our main motivation to develop a new algorithm was not to improve on existing ones for the same problems, but to allow solving new problems: those in which an SDF based formulation is preferred for other reasons, but computing collision detection is necessary. Nevertheless, we need to benchmark our new algorithm, and it is natural to take existing ones as reference.

5.1. Methodology

For testing purposes, we will use some simple 3D geometric shapes as shown in Fig. 7.

To generate the random convex polyhedra we sample uniformly on the surface of the unit sphere and generate the convex hull of the points. We use 50, 200, 500, and 1000 points. Since the SDF algorithm starts with a bounding ball of the polyhedron this method gives it an advantage as they are rounded shapes. To compensate we scale the axes in proportions 3:2:1. These numbers were selected as representative from 52000 shapes in the ShapeNet dataset [34] with median proportions 2.89:1.41:1.

The SDF for the box is exact, taken from [35], while the cone and ellipsoid are approximations using Eq. (21). To evaluate the approximate SDF for polyhedra we use Algorithm 4.

We always consider two shapes of the same type (i.e box vs box). The mean of the bounding sphere radii of the bodies is considered the unit of distance. We generate 1000 random rotations and 100 separation distances uniformly between −0.05 and 0.05. Higher penetrations are unrealistic and higher separations may be resolved with faster broad-phase collision methods. To avoid noise we measure the elapsed time Δt_{100} on 100 evaluations of each instance. To discard outliers [36] we take the minimum of 20 samples:

$$\Delta t = \min_{i=1\ldots20} \frac{\Delta t^i_{100}}{100} \tag{23}$$

All the code to reproduce these benchmarks has been implemented in Julia [37]. For the GJK algorithm, we use [38] slightly modified to speed up the support function evaluation using hill climbing on the vertex adjacency graph. We used our implementation in Julia of the MPR algorithm.

5.2. Collision detection

Since the most predictive parameter for computation time is the object interpenetration we show processing times as a function of the interpenetration in Fig. 8 for the three mentioned shapes and also for a box with rounded edges, as rounding any shape can be easily modeled both in the SDF and the support function.

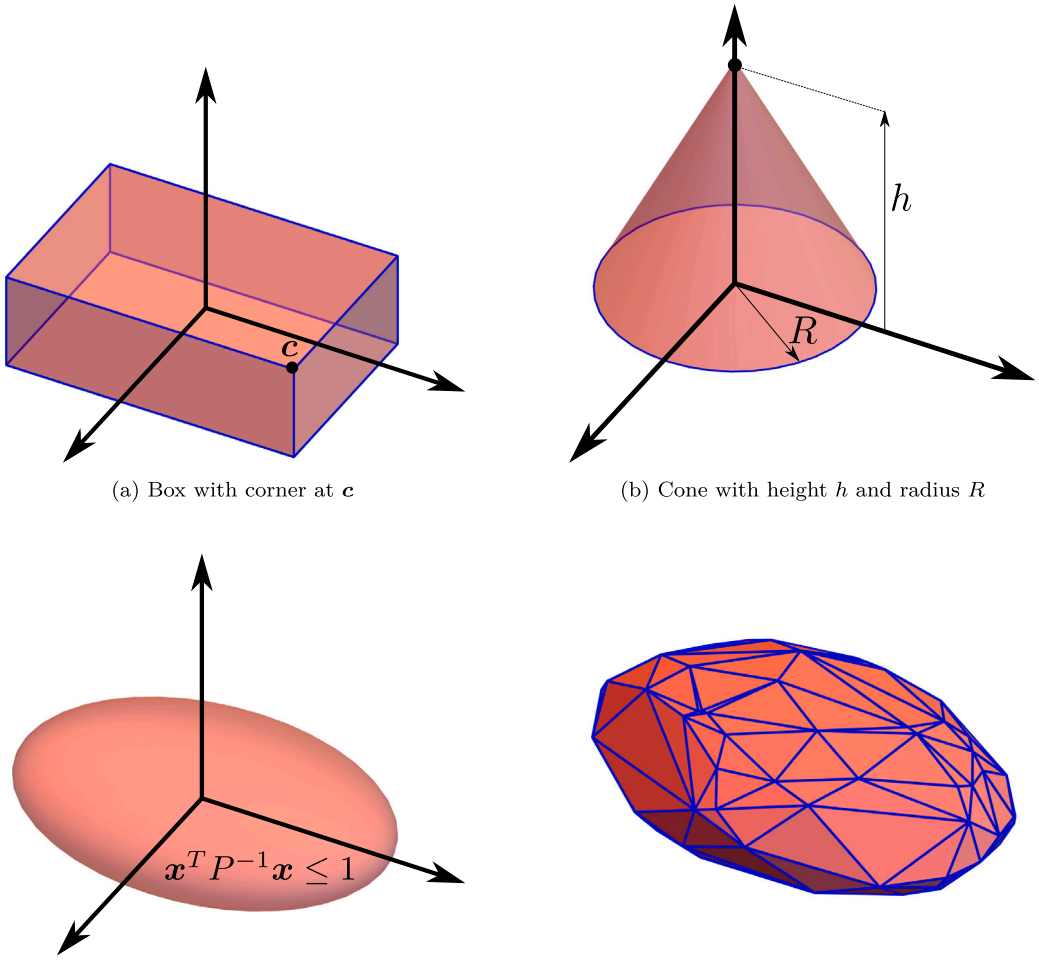
We also compare the time required by GJK vs the time required by SDF and report the percentage of time that SDF ran faster in Fig. 9 and similarly we show MPR vs SDF in Fig. 10

We also report the same results on polyhedra in Figs. 11–13.

As the benchmarks show, for these particular implementations and problem setups, the SDF approach obtains similar results for the collision detection problem. In general simpler shapes tend to favor GJK and MPR even more, while more complex and smoother shapes tend to favor SDF.

5.3. Minimum distance and penetration depth

Since for most of the shapes considered we are using an approximation to the true SDF, our algorithm does not obtain precise results except when the bodies are exactly touching (zero distance). Figs. 14 and 15 show for positive values of distance the difference between the minimum distance as computed by the SDF algorithm and the true minimum distance between the shapes. For negative values there are two sources of error: one is the error introduced by the SDF approximation and the other is not really an error but a difference in the definition of penetration, since the SDF algorithm computes the minimum ball that can be inscribed in the intersection of the bodies



(a) Box with corner at $\boldsymbol{c}$

(b) Cone with height h and radius R

(c) Ellipsoid at the origin with matrix P

(d) Random polyhedron (100 points)

Fig. 7. Benchmarked shapes.

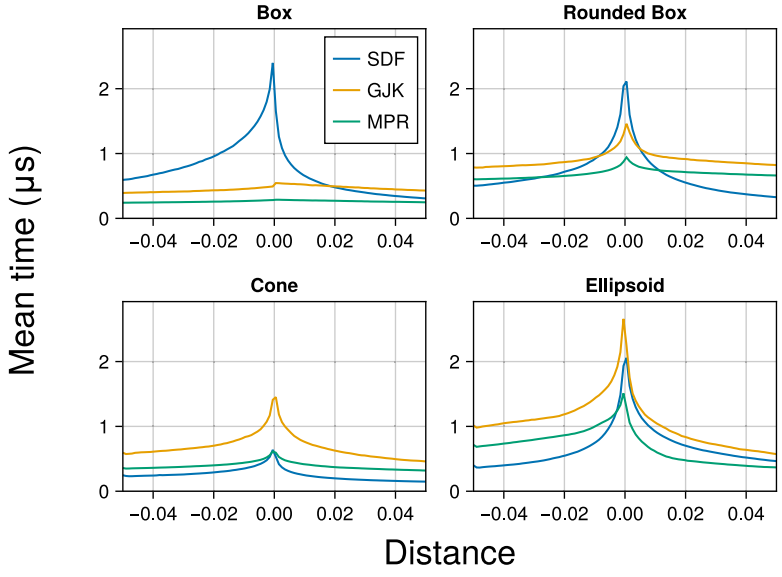


Fig. 8. SDF, GJK and MPR collision detection mean time for simple shapes as a function of distance between the bodies (lower is better). All algorithms have similar results. The spike at distance zero, where the bodies are just touching, is expected as it is the harder case to resolve.

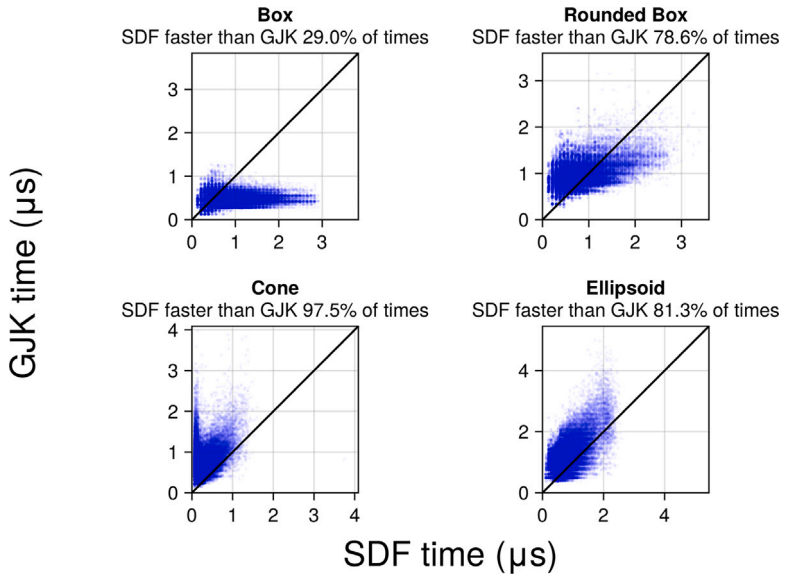


Fig. 9. SDF vs GJK collision detection time for simple shapes. Each point represents a random rotation and separation distance instance. Points falling in the upper triangle represent instances where SDF ran faster than GJK.

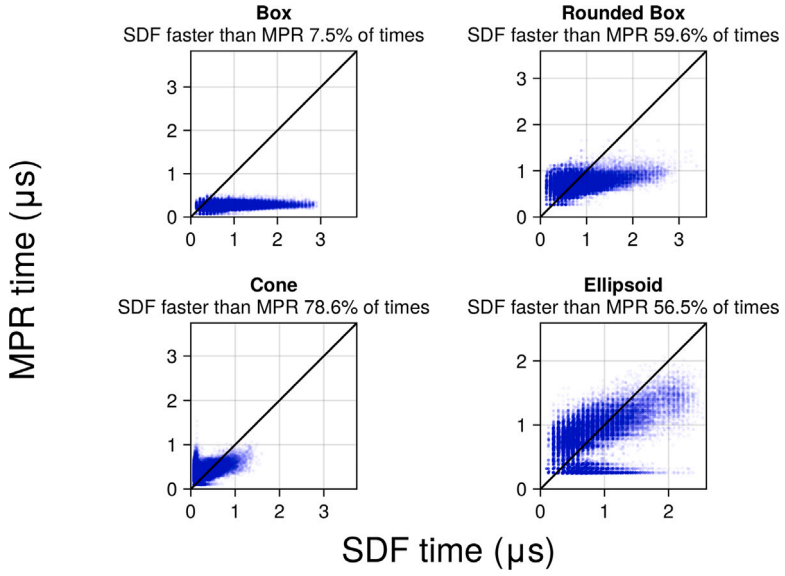


Fig. 10. SDF vs MPR collision detection time for simple shapes. Each point represents a random rotation and separation distance instance. Points falling in the upper triangle represent instances where SDF ran faster than MPR.

(see Fig. 6). Nevertheless, since the translational penetration is the most common definition it is interesting to quantify the difference and we add to the error the difference between the diameter of this ball and the translational penetration. As expected, both the box and bounding box have zero error for non-penetration as the SDF is computed exactly. For penetration there is mismatch between translational penetration and our definition even with exact SDFs, although in rounded shapes like the rounded box, ellipsoid, and polyhedra with higher number of faces, the difference is smaller (as expected). Since MPR also gives an approximation, always an overestimation, to the minimum translational depth, we also plot the MPR error. Since GJK either gives exact distance for positive values or no information for negative values it does not appear on the figure.

In Figs. 16 and 17 we compare the computation time against GJK for positive distances and against MPR for negative distances. We request the same precision for all algorithms, 10^{-6}, and set the maximum number of iterations to 10^4. In our experiments incrementing the number of iterations more does not help in those cases where convergence fails.

The results show that for small polyhedra GJK and MPR are superior to the SDF algorithm, which is unsurprising since they are guaranteed to find exact results in a finite number of iterations. For smooth shapes or polyhedra with a high number of faces the SDF algorithm gets comparable result although we would need to find out how to compute fast enough the exact SDF to compare between exact results on both algorithms. Relative to GJK one advantage of our approach is robustness. Although the GJK algorithm successfully converges for all collision detection instances it fails to converge sometimes to the desired precision of minimum distance. Each red dot on Figs. 16 and 17 represents a GJK convergence failure. GJK convergence failures are treated as outliers and left out of timing computations. Relative to the MPR algorithm we offer, in our opinion, similar implementation simplicity, while also providing minimum distance information. Both algorithms have similar performance in collision detection although for simple shapes MPR computes penetration depth much faster. For more complex shapes our algorithm computes penetration depth at a similar

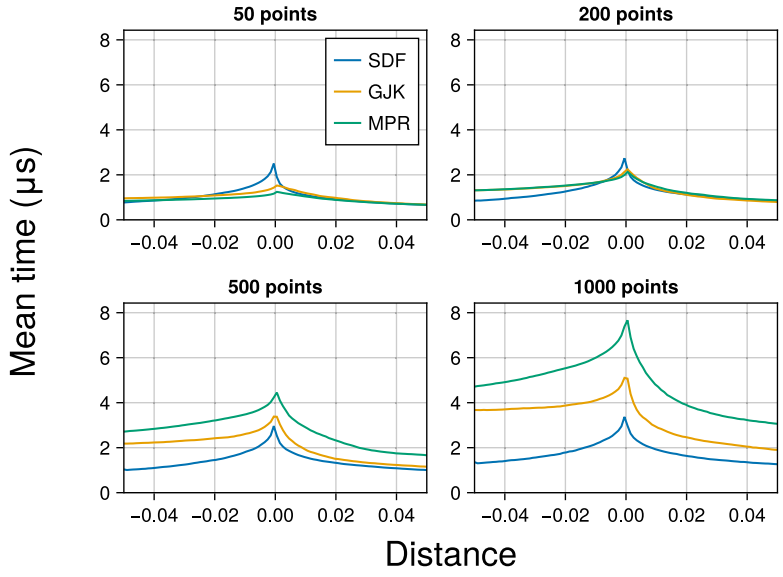


Fig. 11. SDF, GJK and MPR collision detection mean time for polyhedra as a function of distance between the bodies (lower is better). GJK and MPR are faster for simpler shapes and SDF for more complex ones.

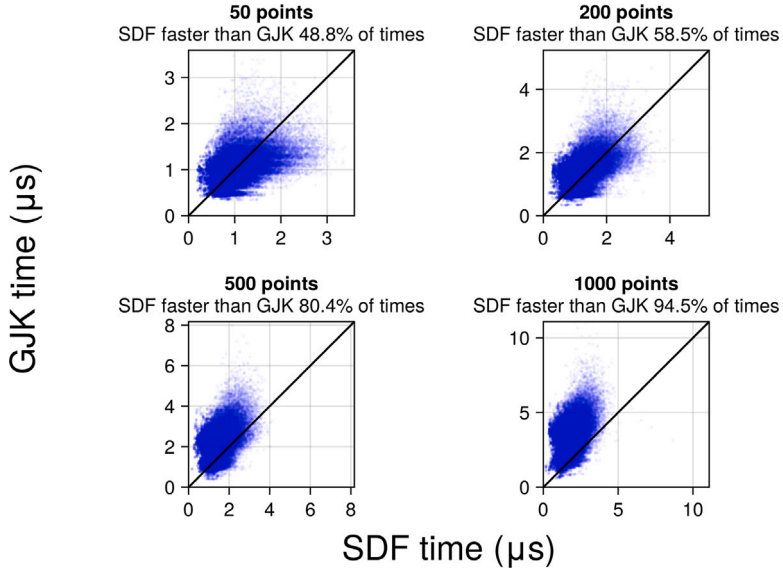


Fig. 12. SDF vs GJK collision detection time for polyhedra. Each point represents a random rotation and separation distance instance. Points falling in the upper triangle represent instances where SDF ran faster than GJK.

speed but with higher precision relative to minimum translational depth.

6. Conclusions and further work

We have developed a simple algorithm based on convex optimization to solve collision detection, minimum distance, and penetration depth when convex shapes are represented via (maybe approximate) SDFs, for arbitrary dimensions. Collision detection is always exactly detected even with approximate SDFs while minimum distance and penetration depth precision depend on computing the exact SDF of the shapes. The main usefulness of our algorithm is the possibility of solving collision problems without converting from the SDF to alternative representations like the support function.

For cases where the support function can also be easily evaluated, the main strengths of the proposed algorithm versus GJK is simplicity (about 100 lines of Julia code) and robustness, since it never failed to converge to the required precision on any instance. Relative to MPR our algorithm returns minimum distance when there is no penetration. Our algorithm is independent, both theoretically and in our implementation, of the number of dimensions although it gets exponentially slower with dimensionality and so it may only be practical in time sensitive applications to the more common case of 2 or 3 dimensions. We have compared its performance against the well-known algorithms GJK and MPR with similar results on collision detection. The GJK algorithm usually performs better on minimum distance and gives exact results while the SDF would require exact SDF evaluations, which may be costly. On the other hand our algorithm gives also penetration depth

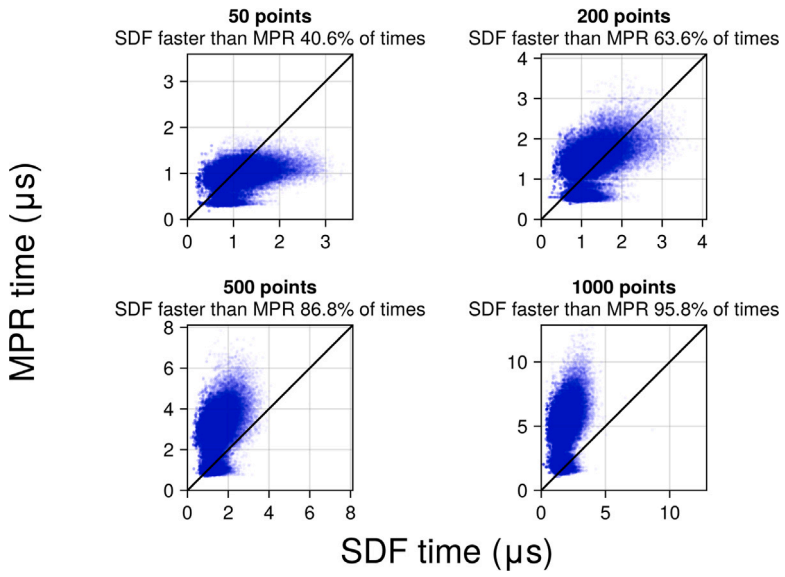


Fig. 13. SDF vs MPR collision detection time for polyhedra. Each point represents a random rotation and separation distance instance. Points falling in the upper triangle represent instances where SDF ran faster than GJK.

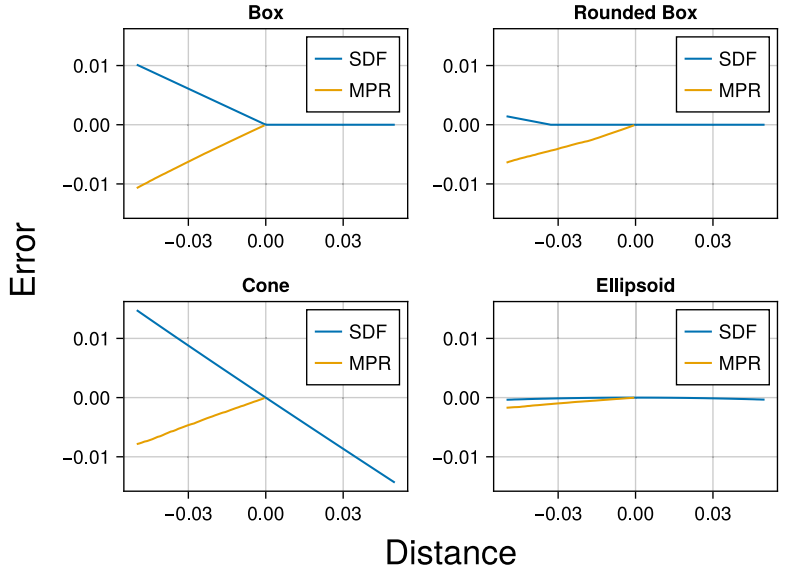


Fig. 14. SDF penetration/minimum distance minus translational penetration/minimum distance for simple shapes. We use exact SDFs for box and rounded box shapes which gives zero error without penetration. With penetration the error is really just the different definition of penetration relative to translational penetration. We use approximate SDFs for the cone and ellipsoid, with quite different error behavior. We also show for comparison results with MPR for negative distance.

and it is more robust. The MPR algorithm usually performs faster on penetration depth although our algorithm usually approximates slightly better the minimum translational depth and returns minimum distance when there is no penetration. It remains future work to compare performance against the Expanding Polytope Algorithm which starts after GJK detects a collision [22].

We have shown some theoretical results regarding incremental problems after body displacements valid also for non-convex shapes. The main advantage of the SDF function is non-convex shape representation and we hope to extend these results in the future to this much more complex case. Since the general non-convex case has exponential complexity this will require suitable restrictions to non-convexity which are left for future work. We hope that our optimization based approach will benefit from results in the non-convex optimization literature.

CRediT authorship contribution statement

Pedro López-Adeva Fernández-Layos: Writing – original draft. **Luis F.S. Merchante:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

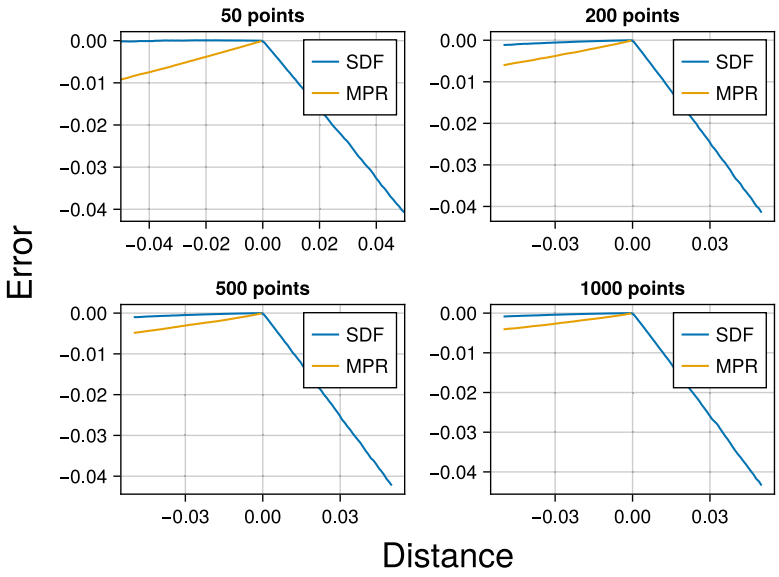


Fig. 15. SDF penetration/minimum distance minus translational penetration/minimum distance for polyhedra. We use approximate SDFs. As the number of points increases the polyhedra gets more rounded and our penetration definition agrees more closely with translational penetration. We also show for comparison results with MPR for negative distance.

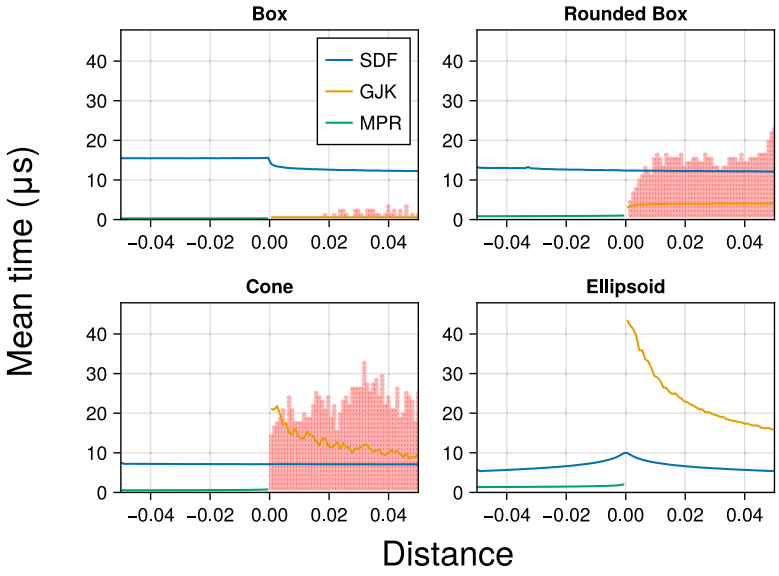


Fig. 16. SDF, GJK and MPR minimum distance/penetration mean time for simple shapes. Except for the box SDF evaluation is approximate and therefore results are approximate while GJK is exact. For simple polyhedral shapes like a box GJK is way faster, but it struggles and even fails to converge for other shapes. Each red point represents an instance (out of 1000 per separation distance) where the GJK algorithm failed to converge to the required absolute precision of 10^{-6} in 10^4 iterations. For these simple shapes MPR gives excellent results, combining robustness with speed and clearly dominates if no minimum distance is required when there is no penetration.

Appendix A. Approximating the support function

We now prove that for any direction we can approximate a support point by projecting back from a point far enough in the given direction (see Fig. A.18). More exactly will prove that as λ increases $p(\lambda \boldsymbol{v})$ gets arbitrarily close to some, but not necessarily fixed, support point $\sigma(\boldsymbol{v})$. Since projections over non-convex bodies are not always unique and σ can be multi-valued even for convex bodies we need to introduce some convergence concepts equivalent to the previous statement.

Definition 1. The distance $d(\boldsymbol{x}, A)$ of a point $\boldsymbol{x}$ to a set A is:

$$d(\boldsymbol{x}, A) = \inf_{\boldsymbol{a} \in A} \|\boldsymbol{x} - \boldsymbol{a}\| \tag{A.1}$$

Definition 2. A sequence $(\boldsymbol{x}_k)_{k\in\mathbb{N}}$ converges to a set A if $d(\boldsymbol{x}_k, A) \to 0$. We write $(\boldsymbol{x}_k)_{k\in\mathbb{N}} \to A$.

Our definition is similar but weaker than Kuratowski convergence for the sequence of sets $(\{\boldsymbol{x}_k\})_{k\in\mathbb{N}}$ [39].

Proposition 7. *If $(\boldsymbol{x}_k)_{k\in\mathbb{N}}$ is a sequence inside a compact set and L its limit set (the set of its accumulation points), then $(\boldsymbol{x}_k)_{k\in\mathbb{N}} \to L$*

Proof. Using the definition of convergence to a set, assume no n exists such that $d(\boldsymbol{x}_k, L) < \varepsilon$ for $k \geq n$ and consider all $k \in K \subset \mathbb{N}$ such that $d(\boldsymbol{x}_k, L) \geq \varepsilon$. Then $(\boldsymbol{x}_k)_{k\in K}$ is an infinite subsequence since otherwise we could pick $n = \max K + 1$ and since it is contained inside a compact set it must have an accumulation point $\boldsymbol{x}'$, but it must be $d(\boldsymbol{x}', L) \geq \varepsilon > 0$ and therefore $\boldsymbol{x}' \notin L$, which is a contradiction. □

Note that if $A \subseteq B$ then also $(\boldsymbol{x}_k)_{k\in\mathbb{N}} \to B$ (in particular $\mathbb{R}^n$). The following simple result gives a lower bound on A. In particular, inside a

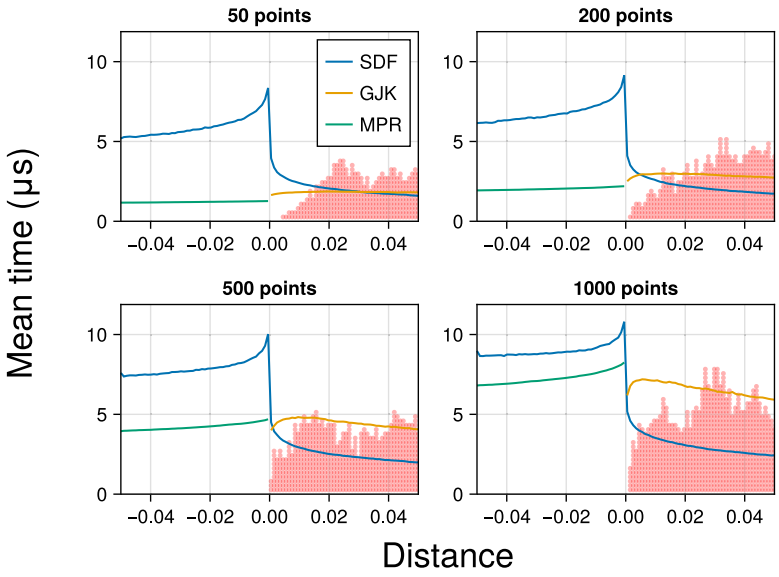


Fig. 17. SDF and GJK minimum distance/penetration mean time for polyhedra. SDF evaluation is approximate and therefore results are approximate while GJK is exact. As the polyhedral shapes get more complex GJK loses its advantage to SDF and starts to fail more often. Each red point represents an instance (out of 1000 per separation distance) where the GJK algorithm failed to converge to the required absolute precision of 10^{-6} in 10^4 iterations. For negative distances MPR is always faster, although it diminishes its advantage with more complex shapes.

compact set the limit set is the smallest closest set a sequence converges to.

Lemma 1. *If L is the limit set of $(\boldsymbol{x}_k)_{k\in\mathbb{N}}$ and $(\boldsymbol{x}_k)_{k\in\mathbb{N}} \to A$ with A closed, then $L \subseteq A$.*

Proof. If $\boldsymbol{x}'$ is an accumulation and $\boldsymbol{x}' \notin A$ then there exists some $\varepsilon > 0$ with $S(\boldsymbol{x}', \varepsilon) \cap A = \emptyset$, containing an infinite subsequence with $d(\boldsymbol{x}_k, A) \geq \varepsilon > 0$ and therefore $(\boldsymbol{x}_k)_{k\in\mathbb{N}} \not\to A$. □

Proposition 8. *For a compact body Ω, any direction $\boldsymbol{v}$, and any sequence $(\lambda_k)_{k\in\mathbb{N}} \to \infty$:*

$$\left(p(\lambda_k \boldsymbol{v})\right)_{k\in\mathbb{N}} \to \sigma(\boldsymbol{v}) \tag{A.2}$$

Proof. We apply the following rewritings to the projection:

$$\begin{aligned} p(\lambda\boldsymbol{v}) &= \operatorname*{argmin}_{\boldsymbol{x}\in\Omega} \|\boldsymbol{x} - \lambda\boldsymbol{v}\| \\ &= \operatorname*{argmin}_{\boldsymbol{x}\in\Omega} \|\boldsymbol{x} - \lambda\boldsymbol{v}\|^2 \\ &= \operatorname*{argmin}_{\boldsymbol{x}\in\Omega} \left(\|\boldsymbol{x}\|^2 + \lambda^2 - 2\lambda(\boldsymbol{x}\cdot\boldsymbol{v})\right) \\ &= \operatorname*{argmin}_{\boldsymbol{x}\in\Omega} \left(\frac{\|\boldsymbol{x}\|^2}{2\lambda} - \boldsymbol{x}\cdot\boldsymbol{v}\right) \end{aligned}$$

Define as a shorthand $g(\boldsymbol{x}) = -\boldsymbol{x}\cdot\boldsymbol{v}$ and $f(\boldsymbol{x}, \lambda) = \frac{\|\boldsymbol{x}\|^2}{2\lambda} - \boldsymbol{x}\cdot\boldsymbol{v}$. Notice that from the definition of the support function $s(\boldsymbol{v}) = \min g(\boldsymbol{x})$.

Since the set Ω is bounded then $\|\boldsymbol{x}\| \leq R$ for some R. We have therefore the following inequalities showing that f converges to g uniformly as $\lambda \to \infty$ on the compact set Ω:

$$g(\boldsymbol{x}) \leq f(\boldsymbol{x}, \lambda) \leq \frac{R^2}{2\lambda} + g(\boldsymbol{x})$$

Uniform convergence on a compact set gives some guarantees on the convergence of the minimizers [40]. We repeat a proof for completeness and simplicity. Taking the minimum over the inequalities:

$$s(\boldsymbol{v}) \leq \min_{\boldsymbol{x}\in\Omega} f(\boldsymbol{x}, \lambda) \leq \frac{R^2}{2\lambda} + s(\boldsymbol{v})$$

For any $(\lambda_k)_{k\in\mathbb{N}} \to \infty$ define $(\boldsymbol{x}_k)_{k\in\mathbb{N}}$ with

$$\boldsymbol{x}_k = p(\lambda_k \boldsymbol{v}) = \operatorname*{argmin}_{\boldsymbol{x}\in\Omega} f(\boldsymbol{x}, \lambda)$$

In case the projection is not unique pick any value. Then as we have seen:

$$s(\boldsymbol{v}) \leq f(\boldsymbol{x}_k, \lambda_k) \leq \frac{R^2}{2\lambda_k} + s(\boldsymbol{v}) \tag{A.3}$$

Since this sequence is inside a compact set it has at least one convergent subsequence $(\boldsymbol{x}_k)_{k\in K} \to \boldsymbol{x}'$ with $K \subset \mathbb{N}$. Defining also $(\lambda_k)_{k\in K}$, substituting in Eq. (A.3) and taking limits on the convergent subsequences we conclude that

$$\lim_{k\to\infty} f(\boldsymbol{x}_k, \lambda_k) = -\boldsymbol{x}'\cdot\boldsymbol{v} = s(\boldsymbol{v}) \implies \boldsymbol{x}' \in \sigma(\boldsymbol{v}) \implies L \subseteq \sigma(\boldsymbol{v})$$

We now simply apply Proposition 7. Since convergence to a subset also gives convergence to the set the proof is finished. □

Collision detection algorithms based on the support function could use Eq. (A.2) as a fallback to approximate the support function if the geometry is specified using the SDF.

Appendix B. Proofs for lemmas and propositions

B.1. Proof for Proposition 1

1. Case $\phi^\star_{AB} \leq 0$.
   Obviously $\phi^\star_{AB} \leq 0 \iff A \cap B \neq \emptyset$. Since for all $\boldsymbol{x} \in A \cap B \implies \phi_{AB}(\boldsymbol{x}) = \phi_{A\cap B}(\boldsymbol{x})$ then $-\phi_{AB}(\boldsymbol{x})$ gives the radius of an inscribed sphere at $\boldsymbol{x} \in A \cap B$, which gets maximized at $\boldsymbol{x}^\star$.
2. Case $\phi^\star_{AB} > 0$.
   First, it must be $\phi_A(\boldsymbol{x}^\star) = \phi_B(\boldsymbol{x}^\star) > 0$. By contradiction, assume $\phi_A(\boldsymbol{x}^\star) < \phi_B(\boldsymbol{x}^\star)$. Since $\phi_B(\boldsymbol{x}^\star) > 0$ (or otherwise $\phi^\star_{AB} \leq 0$) we can move towards $p_B(\boldsymbol{x}^\star)$ to strictly decrease ϕ_B and therefore ϕ_{AB} until $\phi_A = \phi_B$.
   Second, obviously $2\max\left(\phi_A(\boldsymbol{x}), \phi_B(\boldsymbol{x})\right) \geq \phi_A(\boldsymbol{x}) + \phi_B(\boldsymbol{x})$ and since by the first point we can consider only points with $\phi_A(\boldsymbol{x}) > 0$ and $\phi_B(\boldsymbol{x}) > 0$, from the triangle inequality $\phi_A(\boldsymbol{x}) + \phi_B(\boldsymbol{x}) \geq \|p_A(\boldsymbol{x}) - p_B(\boldsymbol{x})\| \geq \|\boldsymbol{x}_A - \boldsymbol{x}_B\|$, where $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ is a pair of closest points between A and B. We have then that $\phi^\star_{AB} \geq \frac{1}{2}\|\boldsymbol{x}_A - \boldsymbol{x}_B\|$ but we can achieve the equality by choosing $x^\star = \frac{1}{2}(\boldsymbol{x}_A + \boldsymbol{x}_B)$. □

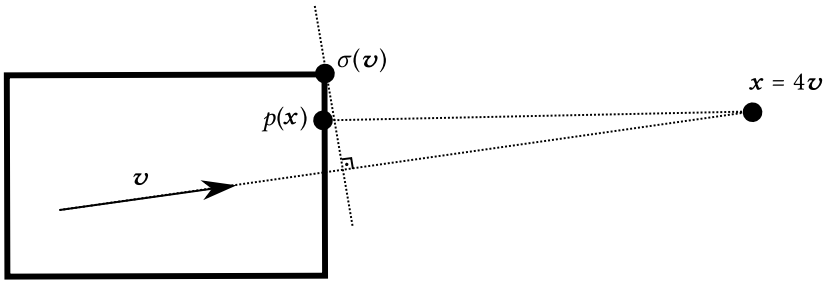


Fig. A.18. The support function σ can be approximated using the SDF. As we project back from a point moving further and further in some direction, the projection converges to a support point. We discard this approach for exact collision detection.

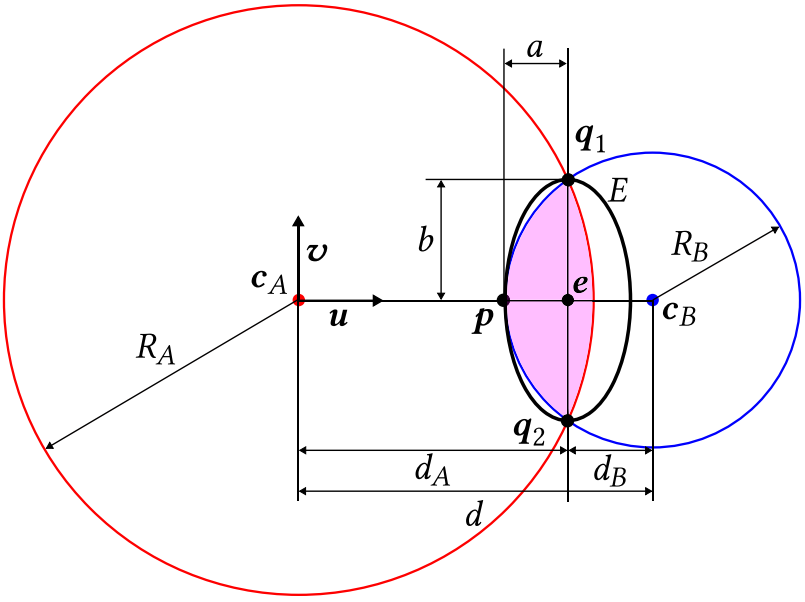


Fig. B.19. Minimum bounding ellipsoid for the intersection of two spheres, computed by Algorithm 3, which can be used as the initial ellipsoid in the ellipsoid algorithm.

B.2. Proof for Proposition 2

We first need the following simple lemma:

Lemma 2. *An ellipsoid with length $2a$ along a unitary vector $\boldsymbol{u}$ and length $2b$ perpendicular to $\boldsymbol{u}$ has matrix $\boldsymbol{P} = \boldsymbol{P}_R(a, b, \boldsymbol{u}) = b^2\boldsymbol{I} + \left(a^2 - b^2\right)\boldsymbol{u}\boldsymbol{u}^T$*

Proof. The ellipsoid has rotational symmetry around $\boldsymbol{u}$ and its points satisfy the equation

$$\frac{1}{a^2}(\boldsymbol{x}\cdot\boldsymbol{u})^2 + \frac{1}{b^2}\left[\|x\|^2 - (\boldsymbol{x}\cdot\boldsymbol{u})^2\right] \leq 1$$

Writing $(\boldsymbol{x}\cdot\boldsymbol{u}) = \boldsymbol{x}^T\left(\boldsymbol{u}\boldsymbol{u}^T\right)\boldsymbol{x}$ and $\|x\|^2 = \boldsymbol{x}^T\boldsymbol{I}\boldsymbol{x}$ we find that $\boldsymbol{Q} = \frac{1}{b^2}\boldsymbol{I} + \left(\frac{1}{a^2} - \frac{1}{b^2}\right)\boldsymbol{u}\boldsymbol{u}^T$ and we can check that the given expression of $\boldsymbol{P}$ is its inverse. □

And now for the proof of the proposition, we follow Fig. B.19 which depicts a two dimensional situation. Higher dimensional problems have rotational symmetry around the axis passing through the ball centers and so can be reduced to two dimensions. Define $\boldsymbol{p}$ as the intersection of $\overline{c_A c_B}$ with ∂S_B, $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ as the intersections of ∂S_A with ∂S_B and $\boldsymbol{e}$ as the intersection of $\overline{c_A c_B}$ with $\overline{q_1 q_2}$.

By simple geometry, the distance $d = \|c_A - c_B\|$ gets partitioned by $\boldsymbol{e}$ as $d = d_A + d_B$ where $d_A = (d + s)/2$ and $d_B = (d - s)/2$ with $s = \left(R_A^2 - R_B^2\right)/d$.

There are three distinct cases depending on d:

1. $d > R_A + R_B$ There is no intersection.
2. $\sqrt{R_A^2 - R_B^2} < d \leq R_A + R_B$. This is the situation depicted in Fig. B.19.
3. $d \leq \sqrt{R_A^2 - R_B^2}$. The resulting ellipsoid is just the smaller ball S_B.

Case (1): strictly speaking any ellipsoid contains the empty set but we choose to raise an error.

Case (2): the minimum bounding ellipse E that contains the points $\boldsymbol{p}$, $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ has major axis $\overline{q_1 q_2}$, center $\boldsymbol{e}$ and semi-minor axis $\overline{ep}$.

We now prove that E not only contains these three points but the chords of ∂S_A and ∂S_B that go between the points $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$, and therefore the region $S_A \cap S_B$: E has at $\boldsymbol{p}$ radius of curvature $R_B + d_B$ and therefore contains ∂S_B in the vicinity of $\boldsymbol{p}$ until they intersect again, but the only remaining intersections are at $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ since there are at most 4 intersections between ∂S_B and E, and the tangent point $\boldsymbol{p}$ already has degree two. E therefore contains the arc of ∂S_B going from $\boldsymbol{q}_1$ to $\boldsymbol{q}_2$. The mirror of this arc through $\overline{q_1 q_2}$ contains the arc of ∂S_A between these two points, which is therefore also contained in E.

Since all ellipses containing $S_A \cap S_B$ must contain also the points $\boldsymbol{p}$, $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ the ellipse E so described is the smallest one containing $S_A \cap S_B$. It has center $\boldsymbol{e} = c_A + d_A\boldsymbol{u}$ and by Lemma 2 matrix $\boldsymbol{P} = \boldsymbol{P}_R\left(R_B - d_B, \sqrt{R_B^2 - d_B^2}, \boldsymbol{u}\right)$.

Case (3): we must return the ball S_B. This can be obtained using the formulae of case (2) for $\boldsymbol{e}$ and $\boldsymbol{P}$ by simply setting $d_A \leftarrow d$ and $d_B \leftarrow 0$. □

B.3. Proof for Proposition 3

Define $\underline{\phi}_{AB}(\boldsymbol{x}) = \max\left(\underline{\phi}_A(\boldsymbol{x}), \underline{\phi}_B(\boldsymbol{x})\right)$. Discarding all $\boldsymbol{x}$ such that $\underline{\phi}_{AB} > \phi^+_{AB}$ is equivalent to keeping only the $\boldsymbol{x}$ such that $\max\left(\underline{\phi}_A(\boldsymbol{x}) - \phi^+_{AB}, \underline{\phi}_B(\boldsymbol{x}) - \phi^+_{AB}\right) \leq 0$, which is the intersection of $S\left(\boldsymbol{c}_A, R_A + \phi^+_{AB}\right)$ and $S\left(\boldsymbol{c}_B, R_B + \phi^+_{AB}\right)$. Using Proposition 2 gives the proposed ellipsoid. □

B.4. Proof for Proposition 5 and Corollary 1

First, we need a simple result for the SDFs of balls.

Lemma 3. *If* $A = S\left(\boldsymbol{c}_A, R_A\right)$ *and* $B = S\left(\boldsymbol{c}_B, R_B\right)$ *then* $\phi^\star_{AB} = \max\Big(-R_A, -R_B, \frac{1}{2}\left[\|\boldsymbol{c}_A - \boldsymbol{c}_B\| - \left(R_A + R_B\right)\right]\Big)$

Proof. The SDFs of the balls are $\phi_A(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{c}_A\| - R_A$ and $\phi_B(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{c}_B\| - R_B$. The expression of $\phi^\star_{AB}$ gives the radius of the biggest inscribed ball in $A \cap B$ if there is an intersection or half the distance between the balls otherwise. □

And now we proceed with the proof of the proposition. In collision, we have $\phi^\star_{AB} \leq 0$ and so

$$A \supseteq S = S\left(\boldsymbol{x}^\star_{AB}, -\phi_A\left(\boldsymbol{x}^\star_{AB}\right)\right)$$
$$B \supseteq T = S\left(\boldsymbol{x}^\star_{AB}, -\phi_B\left(\boldsymbol{x}^\star_{AB}\right)\right)$$

After displacing B we have that

$$B' \supseteq T' = T + \Delta\boldsymbol{x} = S\left(\boldsymbol{x}^\star_{AB} + \Delta\boldsymbol{x}, -\phi_B\left(\boldsymbol{x}^\star_{AB}\right)\right)$$

The SDFs of S and T' are therefore upper bounds of ϕ_A and $\phi_{B'}$ and so $\phi_{AB'}(\boldsymbol{x}) \leq \phi_{ST'}(\boldsymbol{x})$ for all $\boldsymbol{x}$ and minimizing both sides of the inequality $\phi^\star_{AB'} \leq \phi^\star_{ST'}$. We now simply apply Lemma 3 to obtain $\phi^\star_{ST'}$ to get

$$\phi^\star_{AB'} \leq \frac{\phi_A\left(\boldsymbol{x}^\star_{AB}\right) + \phi_B\left(\boldsymbol{x}^\star_{AB}\right) + \|\Delta\boldsymbol{x}\|}{2} \leq \phi^\star_{AB} + \frac{\|\Delta\boldsymbol{x}\|}{2}$$

If A and B are not colliding then calling $\boldsymbol{a} \in A$ and $\boldsymbol{b} \in B$ a pair of closest points we have that $\boldsymbol{b} + \Delta\boldsymbol{x} \in B$ and therefore

$$\phi^\star_{AB'} \leq \frac{\|\boldsymbol{a} - \boldsymbol{b} - \Delta\boldsymbol{x}\|}{2} \leq \frac{\|\boldsymbol{a} - \boldsymbol{b}\| + \|\Delta\boldsymbol{x}\|}{2} = \phi^\star_{AB} + \frac{\|\Delta\boldsymbol{x}\|}{2}$$

We have an upper bound of $\phi^\star_{AB'}$ with and without collision. To find a lower bound notice that since $B = B' + (-\Delta\boldsymbol{x})$ we can apply the upper bound inequality interchanging B and B' to obtain

$$\phi^\star_{AB} \leq \phi^\star_{AB'} + \frac{\|-\Delta\boldsymbol{x}\|}{2} \iff \phi^\star_{AB'} \geq \phi^\star_{AB} - \frac{\|\Delta\boldsymbol{x}\|}{2}$$

The proof of the proposition is finished. To prove the corollary notice that inside the previous proof we directly proved the formula in case of collision. When there is no collision notice that $\|\boldsymbol{b} - \boldsymbol{a}\| = \phi_A(\boldsymbol{x}^\star_{AB}) + \phi_B(\boldsymbol{x}^\star_{AB})$ □

B.5. Proof for Proposition 6

The algorithm maximizes s which is equivalent to minimizing t. To prove that the greedy search through neighboring faces is enough notice that we are applying the well-known algorithm to compute the support of vector $\boldsymbol{x}$ but on the polyhedron with vertices $\boldsymbol{h}_\alpha/\delta_\alpha$, which is the (convex) polar reciprocal of the original polyhedron [41]. □

References

[1] Mamou K. Volumetric hierarchical approximate convex decomposition. In: Lengyel E, editor. In: Game engine gems, vol. 3, Boca Raton: CRC Press; 2016, p. 141–58.

[2] Snethen G. Xenocollide: Complex collision made simple. In: Jacobs S, editor. In: Game programming gems, vol. 7, Australia, Boston, MA: Charles River Media/Course Technology; 2008, p. 165–78.

[3] Gilbert E, Johnson D, Keerthi S. A fast procedure for computing the distance between complex objects in three-dimensional space. IEEE J Robot Autom 1988;4:193–203. http://dx.doi.org/10.1109/56.2083.

[4] Fiser D. LibCCD. 2018, URL: https://github.com/danfis/libccd. [Accessed 05 September 2023].

[5] Coumans E, Bai Y. PyBullet, a Python module for physics simulation for games, robotics and machine learning. 2022, URL: http://pybullet.org/. [Accessed 05 September 2023].

[6] Corporation N. PhysX. 2023, URL: https://github.com/NVIDIA-Omniverse/PhysX. [Accessed 05 September 2023].

[7] Rouwe J. Jolt physics. 2023, URL: https://github.com/jrouwe/JoltPhysics. [Accessed 05 September 2023].

[8] Sherman MA, Seth A, Delp SL. Simbody: Multibody dynamics for biomedical research. Proc. IUTAM 2011;2:241–61. http://dx.doi.org/10.1016/j.piutam.2011.04.023.

[9] Lin M, Canny J. A fast algorithm for incremental distance calculation. In: Proceedings. 1991 IEEE international conference on robotics and automation. Sacramento, CA, USA: IEEE Comput. Soc. Press; 1991, p. 1008–14. http://dx.doi.org/10.1109/ROBOT.1991.131723.

[10] Mirtich B. V-Clip: Fast and robust polyhedral collision detection. ACM Trans Graph 1998;17:177–208. http://dx.doi.org/10.1145/285857.285860.

[11] Jones M, Baerentzen J, Sramek M. 3D distance fields: A survey of techniques and applications. IEEE Trans Vis Comput Graphics 2006;12:581–99. http://dx.doi.org/10.1109/TVCG.2006.56.

[12] Hart JC. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. Vis Comput 1996;12:527–45. http://dx.doi.org/10.1007/s003710050084.

[13] Xu H, Barbič J. Continuous collision detection between points and signed distance fields. In: Workshop on virtual reality interaction and physical simulation. 2014.

[14] Fuhrmann A, Sobottka G, Groß C. Distance fields for rapid collision detection in physically based modeling. In: International conference graphicon. Moscow, Russia, 2003.

[15] Guendelman E, Bridson R, Fedkiw R. Nonconvex rigid bodies with stacking. ACM Trans Graph 2003;22:871–8. http://dx.doi.org/10.1145/882262.882358.

[16] Macklin M, Erleben K, Müller M, Chentanez N, Jeschke S, Corse Z. Local optimization for robust signed distance field collision. Proc. ACM Comput. Graph. Interact. Tech. 2020;3:8:1–8:17. http://dx.doi.org/10.1145/3384538.

[17] Ericson C. Real-time collision detection. CRC Press; 2004.

[18] van den Bergen G. Collision detection in interactive 3D environments. CRC Press; 2003.

[19] Lin MC, Manocha D, Kim YJ. Chapter 39: collision and proximity queries. In: Toth CD, O'Rourke J, Goodman JE, editors. Handbook of discrete and computational geometry. CRC press; 2017.

[20] Cameron S, Culley R. Determining the minimum translational distance between two convex polyhedra. In: 1986 IEEE international conference on robotics and automation proceedings, vol. 3, 1986, p. 591–6. http://dx.doi.org/10.1109/ROBOT.1986.1087645.

[21] Cameron S. Enhancing GJK: Computing minimum and penetration distances between convex polyhedra. In: Proceedings of international conference on robotics and automation, vol. 4, 1997, p. 3112–7. http://dx.doi.org/10.1109/ROBOT.1997.606761, vol. 4.

[22] van der Bergen G. Proximity queries and penetration depth computation on 3D game objects. In: Game developers conference. 2001, URL: https://graphics.stanford.edu/courses/cs468-01-fall/Papers/van-den-bergen.pdf.

[23] Barba L, Langerman S. Optimal detection of intersections between convex polyhedra. In: Proceedings of the 2015 annual ACM-SIAM symposium on discrete algorithms (SODA), proceedings. Society for Industrial and Applied Mathematics; 2014, p. 1641–54. http://dx.doi.org/10.1137/1.9781611973730.109.

[24] Montanari M, Petrinic N, Barbieri E. Improving the GJK algorithm for faster and more reliable distance queries between convex objects. ACM Trans Graph 2017;36:30:1–30:17. http://dx.doi.org/10.1145/3083724.

[25] Krantz SG, Parks HR. Distance to Ck hypersurfaces. J Differential Equations 1981;40:116–20. http://dx.doi.org/10.1016/0022-0396(81)90013-9.

[26] Luo H, Wang X, Lukens B. Variational analysis on the signed distance functions. J Optim Theory Appl 2019;180:751–74. http://dx.doi.org/10.1007/s10957-018-1414-2.

[27] Tagliasacchi A, Delame T, Spagnuolo M, Amenta N, Telea A. 3D Skeletons: a state-of-the-art report. Comput Graph Forum 2016;35:573–97. http://dx.doi.org/10.1111/cgf.12865.

[28] Grötschel M, Lovász L, Schrijver A. The ellipsoid method. In: Grötschel M, Lovász L, Schrijver A, editors. Geometric algorithms and combinatorial optimization, algorithms and combinatorics. Berlin, Heidelberg: Springer; 1993, p. 64–101. http://dx.doi.org/10.1007/978-3-642-78240-4_4.

[29] Boyd S, Boyd SP, Vandenberghe L. Convex optimization. Cambridge University Press; 2004.

[30] Ricci A. A constructive geometry for computer graphics. Comput J 1973;16:157–60. http://dx.doi.org/10.1093/comjnl/16.2.157.

[31] Shapiro V. Semi-analytic geometry with R-functions. Acta Numer 2007;16:239–303. http://dx.doi.org/10.1017/S096249290631001X.

[32] Faure F, Barbier S, Allard J, Falipou F. Image-based collision detection and response between arbitrary volume objects. In: Proceedings of the 2008 ACM SIGGRApH/eurographics symposium on computer animation. SCA '08, Goslar, DEU: Eurographics Association; 2008, p. 155–62.

[33] Weller R, Zachmann G. A unified approach for physically-based simulations and haptic rendering. In: Proceedings of the 2009 ACM SIGGRApH symposium on video games. Sandbox '09, New York, NY, USA: Association for Computing Machinery; 2009, p. 151–9. http://dx.doi.org/10.1145/1581073.1581097.

[34] Chang AX, Funkhouser T, Guibas L, Hanrahan P, Huang Q, Li Z, et al. ShapeNet: An information-rich 3d model repository. 2015, URL: http://arxiv.org/abs/1512.03012, arXiv:1512.03012.

[35] Quilez I. Inigo quilez. 2016, URL: https://iquilezles.org/articles/distfunctions/. [Accessed 18 July 2023].

[36] Revels J. Linux-based environments · BenchmarkTools .jl. 2023, URL: https://JuliaCI.github.io/BenchmarkTools.jl/linuxtips/.[Accessed 28 October 2023].

[37] Bezanson J, Edelman A, Karpinski S, Shah VB. Julia: A fresh approach to numerical computing. SIAM Rev 2017;59:65–98. http://dx.doi.org/10.1137/141000671.

[38] Lakshmanan A. ConvexBodyProximityQueries. 2022, URL: https://github.com/arlk/ConvexBodyProximityQueries.jl. [Accessed 18 July 2023].

[39] Berger M, De La Harpe P, Hirzebruch F, Hitchin NJ, Hörmander L, Kupiainen A, et al., editors. Painlevé-Kuratowski Convergence. Grundlehren Der Mathematischen Wissenschaften, vol. 317, Berlin, Heidelberg: Springer; 1998, p. 111–6, http://dx.doi.org/10.1007/978-3-642-02431-3.

[40] Kall P. Approximation to optimization problems: An elementary review. Math Oper Res 1986;11:9–18.

[41] Polarity of polytopes and polyhedral sets. Graduate texts in mathematics, vol. 90, New York, NY: Springer; 1983, p. 56–62. http://dx.doi.org/10.1007/978-1-4612-1148-8.