Differentiable Collision Detection: a Randomized Smoothing Approach

Louis Montaut, Quentin Le Lidec, Antoine Bambade, Vladimír Petrík, Josef Sivic, Justin Carpentier

Differentiable Collision Detection: a Randomized Smoothing Approach

Louis Montaut, Quentin Le Lidec, Antoine Bambade, Vladimir Petrik, Josef Sivic and Justin Carpentier

***Abstract*— Collision detection appears as a canonical operation in a large range of robotics applications from robot control to simulation, including motion planning and estimation. While the seminal works on the topic date back to the 80s, it is only recently that the question of properly differentiating collision detection has emerged as a central issue, thanks notably to the ongoing and various efforts made by the scientific community around the topic of differentiable physics. Yet, very few solutions have been suggested so far, and only with a strong assumption on the nature of the shapes involved. In this work, we introduce a generic and efficient approach to compute the derivatives of collision detection for *any* pair of convex shapes, by notably leveraging randomized smoothing techniques which have shown to be particularly adapted to capture the derivatives of non-smooth problems. This approach is implemented in the HPP-FCL and Pinocchio ecosystems, and evaluated on classic datasets and problems of the robotics literature, demonstrating few micro-second timings to compute informative derivatives directly exploitable by many real robotic applications including differentiable simulation.**

I. Introduction

Collision detection is a crucial stage for many robotic applications including motion planning, trajectory optimization, or simulation. It is also used in many other related domains, such as computer graphics or computational geometry, just to name a few. In particular for simulation, collision detection is a central component of any physical simulator, allowing to assess the collision between two geometries, to retrieve the contact regions if any, or to compute the closest points between the shapes (also known as *witness points* [1]). In addition to the evaluation of collisions, many problems in robotics also rely on the derivatives of geometric contact information such as optimal shape design, grasp synthesis, or differentiable simulation.

Over the past few years, differentiable simulation has gained interest within the robotics and machine learning communities thanks to the progress in gradient-based optimization techniques. Differentiable simulation consists in evaluating the gradient (or sub-gradient in case of a non-smooth contact interaction) of the simulation steps, which can then be exploited by any gradient-based optimization algorithm to efficiently solve various robotic problems involving contact interactions [2], [3], [4], [5], [6], [7]. However, most of the existing works on the topic have only considered the differentiation of the physical principles (Coulomb friction constraints, maximum dissipation principles, Signorini conditions, etc.), without systematically considering the contribution of collision detection in the derivatives, for example, by limiting themselves to a specific type of shapes [4], [7].

To the best of our knowledge, no systematic procedure to compute collision derivatives has been proposed in the literature and it remains a challenging problem. In this paper, we propose to address this issue by introducing a generic approach to compute gradient information of collision detection instances in the context of convex shapes (not necessarily strictly convex), without assuming any regularity on the shapes. This includes not only the standard geometric primitives (ellipsoid, sphere, cone, cube, cylinder, capsules, etc.) but also meshes, which are the standard representation of objects in many applications involving physics simulation and a wide range of robotic applications. Our key contributions lie in leveraging randomized smoothing techniques [8] to estimate the gradients of the collision detection procedure through two proposed estimators:

- a $0^{\text{th}}-$order estimator (see Fig. 1), which does not make any assumption on the nature of the collision algorithms in use,
- a $1^{\text{st}}-$order estimator (see Fig. 2), which exploits the optimality conditions of the underlying optimization program inherent to collision detection with convex shapes. Remarkably, this estimator is at the same time more accurate and less expensive to derive than the 0^{th}-order estimator.

After stating the problem of collision detection in Sec. II, we evaluate the efficiency of these two derivative estimators introduced in Sec. III and IV on standard optimization problems (Sec. V). We will release our code as open-source within the Pinocchio [9] framework, so that it can be easily disseminated into other existing robotic software and applications.

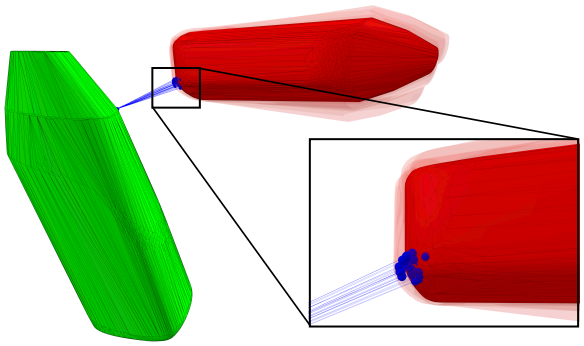


Fig. 1: **Illustration of randomized smoothing approximation on two meshes from the YCB dataset. 0^{th}-order method, Gaussian distribution, $M = 25$ samples.** The witness points for each perturbed relative pose are drawn on both objects. Randomized smoothing allows to recover information regarding the curvature of the underlying objects.

II. Problem statement

Collision detection is a very old topic within the robotics community. It consists in determining whether a pair of

shapes are in collision or not, and in finding the contact point locations [1], [10], [11], [12]. To lower the computational burden, the shapes are often assumed to be convex or decomposed as a collection of convex meshes [13]. Such convexity assumptions have led to highly efficient, generic and robust algorithms to solve collision detection problems, most of which belong to the family of the Gilbert-Johnson-Keerthi (GJK) algorithms [10], [11], [14], [12].

From an optimization perspective, the problem of collision detection can be generically formulated as a convex minimization program of the form:

$$\boldsymbol{x}_1^*, \boldsymbol{x}_2^* = \underset{\boldsymbol{x}_1 \in \mathcal{A}_1,\ \boldsymbol{x}_2 \in \mathcal{A}_2}{\arg\min} \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2, \tag{1}$$

where $\|\cdot\|_2$ is the Euclidean norm, $\mathcal{A}_1$ and $\mathcal{A}_2$ are convex shapes and $\boldsymbol{x}_1^*$ and $\boldsymbol{x}_2^*$ are the so-called witness points. The GJK algorithm is typically used to solve (1). When the objects are in collision, most physics simulations extend (1) to also recover the *penetration depth*, in order, for instance, to correct the interpenetration between rigid objects, inherent to the discrete integration scheme used inside physical simulators. The penetration depth corresponds to the length of the smallest translation which must be applied on the relative configuration between $\mathcal{A}_1$ and $\mathcal{A}_2$ in order to separate them. To compute this quantity, the Expanded Polytope Algorithm (EPA) [1] is typically used and enables the computation of witness points $\boldsymbol{x}_1^*$ and $\boldsymbol{x}_2^*$ in the case of penetration. In both penetration and non-collision cases, the witness points computed by GJK and EPA lie on the boundaries of the objects considered. Finally, in the presence or absence of collision, the signed distance between $\mathcal{A}_1$ and $\mathcal{A}_2$ is defined as:

$$d(\mathcal{A}_1, \mathcal{A}_2) = \begin{cases} \|\boldsymbol{x}_1^* - \boldsymbol{x}_2^*\| & \text{if } \mathcal{A}_1 \cap \mathcal{A}_2 = \emptyset, \\ -\|\boldsymbol{x}_1^* - \boldsymbol{x}_2^*\| & \text{otherwise.} \end{cases} \tag{2}$$

To parameterize the relative position and orientation of $\mathcal{A}_1$ and $\mathcal{A}_2$ in the 3D space, it is convenient to consider their relative configuration, $T(\boldsymbol{q}) \in SE(3)$ which is parameterized by a vector $\boldsymbol{q} \in \mathbb{R}^7$ (3 coordinates for translation and 4 for rotation, encoded by a quaternion). From a mathematical perspective, the main objective of this paper is to retrieve the partial derivatives of (1) w.r.t $\boldsymbol{q}$ namely $\partial \boldsymbol{x}_1^*/\partial \boldsymbol{q}$ and $\partial \boldsymbol{x}_2^*/\partial \boldsymbol{q}$, in the two cases where the two geometries are in collision or not. It is worth mentioning at this stage that $\partial \boldsymbol{x}_1^*/\partial \boldsymbol{q}$ and $\partial \boldsymbol{x}_2^*/\partial \boldsymbol{q}$ are Jacobian matrices of dimension 3×6, with $\partial \boldsymbol{q}$ being an element of the tangent of $SE(3)$ in $T(\boldsymbol{q})$ [15].

Finite differences and other existing methods are likely to fail at capturing informative gradients in most cases. Indeed, even if an object is originally smooth, its approximation as a mesh may present some non smoothness properties (typically localised around the vertices of the mesh), which are difficult to handle for classic optimization-based techniques. More precisely, because they are locally flat or non-smooth, the resulting gradients tend to *"forget"* the curvature of the original surface they approximate [7], leading to uninformative gradients. To overcome these issues, we introduce hereafter two approaches to retrieve a precise estimator of the partial derivatives $\partial \boldsymbol{x}_1^*/\partial \boldsymbol{q}$ and $\partial \boldsymbol{x}_2^*/\partial \boldsymbol{q}$ which are extensively cross-validated against finite differences in Sec. V.

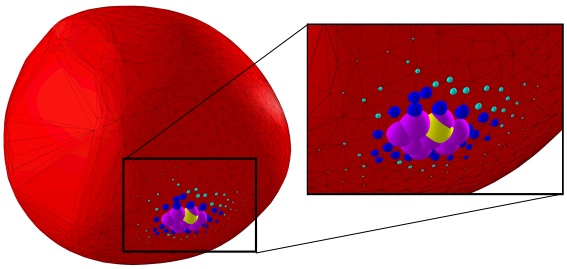


Fig. 2: **Soft-max weights of the Gumbel distribution for the 1^{st}-order gradient estimator, mesh from the YCB dataset.** Different colors represent neighbors of different levels. In yellow: the current-witness point of the object. In purple: level 1 neighbors. In blue: level 3 neighbors. In cyan: level 5 neighbors. The size of the neighbors is proportional to the soft-max weight.

III. 0^{TH}-ORDER ESTIMATION OF COLLISION DETECTION DERIVATIVES

In this section, we introduce a generic approach to compute a 0^{th}-order estimate of the collision detection derivatives by leveraging randomized smoothing techniques which we first review.

A. Background on randomized smoothing

Originally used in black-box optimization algorithms [16], [17], randomized smoothing was recently introduced in the machine learning community [18], [8] in order to integrate discrete operations inside neural networks.

In more details, any function g can be approximated by convolving it with a probability distribution μ:

$$g_\epsilon(x) = \mathbb{E}_{Z \sim \mu}\left[g(x + \epsilon Z)\right], \tag{3}$$

which corresponds to the randomly smoothed counterpart of g and which can be estimated with a Monte-Carlo estimator as follows:

$$g_\epsilon(x) \approx \frac{1}{M} \sum_{i=0}^{M} g(x + \epsilon z^{(i)}), \tag{4}$$

where $\{z^{(1)}, \ldots, z^{(M)}\}$ are i.i.d. samples and M is the number of samples. The ϵ parameter controls the level of noise injected in g_ϵ. Intuitively, the convolution makes g_ϵ smoother than its original counterpart g and, thus, yields better conditioned gradients [8].

Using an integration by part yields a 0^{th}-order estimator of the gradient:

$$\nabla_x^{(0)} g_\epsilon(x) = \frac{1}{M} \sum_{j=0}^{M} -g(x + \epsilon z^{(j)}) \frac{\nabla \log \mu(z^{(j)})^\top}{\epsilon}. \tag{5}$$

This 0^{th}-order estimator can be used even when g is non-differentiable to obtain first-order information, which can then be exploited by any gradient-based optimization technique, as shown by applications in machine learning [8], computer vision [19], [20] and robotics for the optimal control of non-smooth dynamical systems [21], [22].

B. Direct application of randomized smoothing to collision detection

Randomized smoothing can be applied to collision detection by considering the witness points as functions of the configuration vector, i.e., $(\boldsymbol{x}_1^*(\boldsymbol{q}), \boldsymbol{x}_2^*(\boldsymbol{q}))$, regardless of the method used to compute them. We choose to use the combination of the two complementary and state-of-the-art algorithms, the Gilbert, Johnson and Keerthi algorithm (GJK) [10] and Expanding Polytope Algorithm (EPA) [11], to handle collision detection including penetration. Given two convex shapes $\mathcal{A}_1$ and $\mathcal{A}_2$ and a relative pose $T(\boldsymbol{q})$, both of these algorithms allow to compute a pair of witness points $(\boldsymbol{x}_1^*(\boldsymbol{q}), \boldsymbol{x}_2^*(\boldsymbol{q}))$.

To compute a 0^{th}-order estimator of the gradients of (1), we perturb M times the configuration vector by sampling from a distribution μ:

$$\nabla_{\boldsymbol{q}}^{(0)} \boldsymbol{x}_{i,\epsilon}^*(\boldsymbol{q}) = \frac{1}{M} \sum_{j=0}^{M} -\boldsymbol{x}_i^*(\boldsymbol{q} + \epsilon \boldsymbol{z}^{(j)}) \frac{\nabla \log \mu(\boldsymbol{z}^{(j)})^\top}{\epsilon}, \quad (6)$$

for $i = 1, 2$. As a consequence, this 0^{th}-order estimator requires to run the GJK+EPA procedure M times. While the choice of distribution may lead to different convolution effects [19], [20], we found that the standard Gaussian distribution is well adapted to adequately sample over $SE(3)$. In Fig. 1, we give an intuition of the smoothing which results from the 0^{th}-order estimator. To keep the visualization simple, we fix the pose of $\mathcal{A}_1$ (in green) and perturb the pose of $\mathcal{A}_2$. The cloud of resulting witness-points captures the local geometry of $\mathcal{A}_2$. Finally, note that finite differences is a sub-case of randomized smoothing, which considers a specific non-smooth distribution. By construction, finite differences capture less of the underlying geometry, as it deterministically defines the sampling directions and size of the steps taken in such directions. These fundamental differences are quantitatively illustrated in Table 5 of Sec. V.

IV. 1^{ST}-ORDER ESTIMATION OF COLLISION DETECTION DERIVATIVES

In this section, we exploit the optimality conditions of the collision detection problem in order to introduce a computationally efficient and generic approach to derive a 1^{st}-order estimate of the collision detection derivatives. Similarly to the 0^{th}-order approach developed in Sec. III, we leverage randomized smoothing to compute the local Hessian information around the witness points required in the computation of specific gradient quantities. In the particular case of meshes, we notably introduce a closed-form Hessian expression which can be directly computed from the vertices located in the neighborhood of the witness points.

The GJK paradigm for collision detection. To handle the problem of collision detection, we put ourselves in the paradigm of GJK [10], [12], to which EPA [11] also belongs. In this paradigm, the focus is set on computing the *separation vector* between $\mathcal{A}_1$ and $\mathcal{A}_2$. The separation vector is defined as the smallest translation which must be applied to the relative configuration between $\mathcal{A}_1$ and $\mathcal{A}_2$ in order to (i) bring the shapes into collision if they are not in collision, or (ii) separate the shapes if they are in collision. Whether or not the shapes are in collision, the separation vector *always* satisfies the following equation:

$$\boldsymbol{x}^* = \boldsymbol{x}_1^* - \boldsymbol{x}_2^*, \quad (7)$$

where $\boldsymbol{x}_1^*$ and $\boldsymbol{x}_2^*$ are the witness points, obtained as a by-product of the GJK and EPA algorithms.

Although this change in paradigm seems anecdotal, the problem of computing the separation vector $\boldsymbol{x}^*$ between $\mathcal{A}_1$ and $\mathcal{A}_2$ is conveniently fully encapsulated in a minimization problem over the Minkowski difference of the two shapes $\mathcal{D} = \mathcal{A}_1 - \mathcal{A}_2 = \{\boldsymbol{x} = \boldsymbol{x}_1 - \boldsymbol{x}_2 \mid \boldsymbol{x}_1 \in \mathcal{A}_1, \boldsymbol{x}_2 \in \mathcal{A}_2\}$:

$$\begin{aligned} \boldsymbol{x}^* = &\arg\min \|\boldsymbol{x}\|_2^2 \\ &\text{s.t. } \boldsymbol{x} \in \delta\mathcal{D}, \end{aligned} \quad (8)$$

where $\delta\mathcal{D}$ is the boundary of the Minkowski difference. Remarkably, the shapes are in collision if and only if the origin lies inside the Minkowski difference, i.e., $\mathbf{0} \in \mathcal{D}$, as shown in the seminal work of [10] and revisited in [12]. In summary, the separation vector is obtained by projecting the origin onto the surface of the Minkowski difference.

Optimality conditions of collision detection. In order to solve (8), GJK and EPA both solve a sequence of simple linear programming sub-problems. Each sub-problem consists in computing the so-called *support function* of $\mathcal{D}$:

$$\sigma_{\mathcal{D}}(\boldsymbol{x}) = \max_{\boldsymbol{y} \in \mathcal{D}} \langle \boldsymbol{y}, \boldsymbol{x} \rangle, \quad (9)$$

where $\boldsymbol{x} \in \mathbb{R}^3$, $\boldsymbol{y} \in \mathcal{D}$ and $\langle \cdot, \cdot \rangle$ is the Euclidian dot-product. A support point $\boldsymbol{s} \in \delta\mathcal{D}$ belongs to the support set $\partial\sigma_{\mathcal{D}}(\boldsymbol{x})$, corresponding to the sub-gradient of $\sigma_{\mathcal{D}}(\boldsymbol{x})$, if and only if, it is a maximizer of (9). Such a point always exists and is *always* a point on the boundary of the Minkowski difference.

Computing $\sigma_{\mathcal{D}}(\boldsymbol{x})$ corresponds to minimizing a linearization of the objective function of (8) at point $-\boldsymbol{x}$. Remarkably, when it is not possible to find a point $\boldsymbol{x} \in \mathcal{D}$ which further decreases this linearization, both GJK and EPA have reached the optimal solution $\boldsymbol{x}^*$, as Pb. (8) is convex. This optimality condition corresponds to the convergence criterion of both GJK [10] and EPA [11] and can be stated as follows:

$$\begin{cases} \boldsymbol{x}^* \in \partial\sigma_{\mathcal{D}}(-\boldsymbol{x}^*) & \text{if } \mathbf{0} \notin \mathcal{D}, \\ \boldsymbol{x}^* \in \partial\sigma_{\mathcal{D}}(\boldsymbol{x}^*) & \text{otherwise.} \end{cases} \quad (10)$$

Eq. (10) is directly linked to the Frank-Wolf duality-gap, a convergence criteria which allows to measure the progress towards an optimal solution. We refer to [12] for a complete analysis. Eq. (10) is handy as it corresponds to the optimality condition of (8) and characterizes $\boldsymbol{x}^*$. Note that we choose to compute $\boldsymbol{x}^*$ using GJK and EPA but (10) is true however $\boldsymbol{x}^*$ is obtained. For the sake of simplicity, we will assume $\mathbf{0} \notin \mathcal{D}$ but the rest of this section is applicable to the case where $\mathbf{0} \in \mathcal{D}$.

For now, let us suppose first that $\mathcal{D}$ is smooth and strictly-convex, which corresponds to the case where the shapes $\mathcal{A}_1$ and $\mathcal{A}_2$ are smooth and strictly-convex (i.e., ellipsoids or

spheres). The support set is reduced to a singleton for any $\boldsymbol{x}$: $\partial\sigma_{\mathcal{D}}(\boldsymbol{x}) = \nabla\sigma_{\mathcal{D}}(\boldsymbol{x})$ where $\nabla\sigma_{\mathcal{D}}$ is the gradient of the support function, i.e., the only maximizer of (9). In such a case, (10) reduces to an equality.

In practice, we use the support functions of the shapes denoted $\sigma_{\mathcal{A}_1}$ and $\sigma_{\mathcal{A}_2}$ to compute $\sigma_{\mathcal{D}}$, as $\sigma_{\mathcal{D}} = \sigma_{\mathcal{A}_1} - \sigma_{\mathcal{A}_2}$ [10], [12]. Since we consider the relative pose $T(\boldsymbol{q})$ between $\mathcal{A}_1$ and $\mathcal{A}_2$, all vectors are classically expressed in the frame of $\mathcal{A}_1$. By decomposing the terms in the support function, we can show that for any $\boldsymbol{x}$, if $\boldsymbol{s}_1 \in \partial\sigma_{\mathcal{A}_1}(\boldsymbol{x})$ and $\boldsymbol{s}_2 \in \partial\sigma_{\mathcal{A}_1}(-R(\boldsymbol{q})^T\boldsymbol{x})$, then:

$$\boldsymbol{s} = \boldsymbol{s}_1 - \boldsymbol{s}_2 \in \partial\sigma_{\mathcal{D}}(\boldsymbol{x}), \tag{11}$$

where $R(\boldsymbol{q})$ is the rotation matrix associated to $T(\boldsymbol{q})$.

In practice, evaluating and finding a maximizer of the support function is simple and a computationally cheap operation (this partly explains the large popularity of GJK and related algorithms for collision detection [1]). Since this is true also for $\boldsymbol{x}^*$, we rewrite Eq. (10) to obtain:

$$\boldsymbol{x}^* - \nabla\sigma_{\mathcal{A}_1}(-\boldsymbol{x}^*) + T(\boldsymbol{q})\nabla\sigma_{\mathcal{A}_2}(R(\boldsymbol{q})^T\boldsymbol{x}^*) = \boldsymbol{0}. \tag{12}$$

Remarkably, the witness points $\boldsymbol{x}_1^*$ and $\boldsymbol{x}_2^*$ appear in (12):

$$\begin{cases} \boldsymbol{x}_1^*(\boldsymbol{x}^*) = \nabla\sigma_{\mathcal{A}_1}(-\boldsymbol{x}^*) \\ \boldsymbol{x}_2^*(\boldsymbol{x}^*, \boldsymbol{q}) = T(\boldsymbol{q})\nabla\sigma_{\mathcal{A}_2}(R(\boldsymbol{q})^\top\boldsymbol{x}^*). \end{cases} \tag{13}$$

Implicit function differentiation. We define the function $f : \mathbb{R}^3 \times \mathbb{R}^7 \to \mathbb{R}^3$ as:

$$f(\boldsymbol{x}, \boldsymbol{q}) = \boldsymbol{x} - \nabla\sigma_{\mathcal{A}_1}(-\boldsymbol{x}) + T(\boldsymbol{q})\nabla\sigma_{\mathcal{A}_2}(R(\boldsymbol{q})^T\boldsymbol{x}). \tag{14}$$

From (12), the separation vector $\boldsymbol{x}^*(\boldsymbol{q})$, parameterized by $\boldsymbol{q}$, is thus implicitly described by the equation:

$$f(\boldsymbol{x}^*, \boldsymbol{q}) = \boldsymbol{0}. \tag{15}$$

By expanding the 1st-order terms of f, we can relate the sensitivity of $\boldsymbol{x}^*$ to the relative configuration $\boldsymbol{q}$ between the shapes:

$$\frac{\partial f(\boldsymbol{x}^*, \boldsymbol{q})}{\partial \boldsymbol{x}^*}\delta\boldsymbol{x}^* + \frac{\partial f(\boldsymbol{x}^*, \boldsymbol{q})}{\partial \boldsymbol{q}}\delta\boldsymbol{q} = \boldsymbol{0}, \tag{16}$$

leading to the following relation:

$$\frac{\partial f(\boldsymbol{x}^*, \boldsymbol{q})}{\partial \boldsymbol{x}^*}\frac{\partial \boldsymbol{x}^*}{\partial \boldsymbol{q}} = -\frac{\partial f(\boldsymbol{x}^*, \boldsymbol{q})}{\partial \boldsymbol{q}}, \tag{17}$$

and finally to:

$$\frac{\partial \boldsymbol{x}^*}{\partial \boldsymbol{q}} = -\left[\frac{\partial f(\boldsymbol{x}^*, \boldsymbol{q})}{\partial \boldsymbol{x}^*}\right]^{-1}\frac{\partial f(\boldsymbol{x}^*, \boldsymbol{q})}{\partial \boldsymbol{q}}, \tag{18}$$

if the Jacobian of f w.r.t. $\boldsymbol{x}^*$ is invertible, where:

$$\frac{\partial f(\boldsymbol{x}^*, \boldsymbol{q})}{\partial \boldsymbol{x}^*} = I + \frac{\partial^2\sigma_{\mathcal{A}_1}(-\boldsymbol{x}^*)}{\partial \boldsymbol{x}^{*2}} + R(\boldsymbol{q})\frac{\partial^2\sigma_{\mathcal{A}_2}(\boldsymbol{y}^*)}{\partial \boldsymbol{y}^{*2}}R(\boldsymbol{q})^\top, \tag{19}$$

with $\boldsymbol{y}^* = R(\boldsymbol{q})\boldsymbol{x}^*$. The terms in $\partial f(\boldsymbol{x}^*, \boldsymbol{q})/\partial \boldsymbol{q}$ are derivative terms involving elements of $SE(3)$ and can be simply obtained by following the derivations in [15].

To evaluate both derivative terms of f, it is necessary to compute the Hessian of the support function $\partial^2\sigma_{\mathcal{A}}(\boldsymbol{x})/\partial\boldsymbol{x}^2$ at $\boldsymbol{x}^*$ (or $\boldsymbol{y}^*$) where $\mathcal{A}$ stands for either shape $\mathcal{A}_1$ or $\mathcal{A}_2$. The Hessian of the support function encodes the local curvature of the shape and it is easy to compute for basic smooth and strictly-convex shapes such as spheres or ellipsoids.

Let us now focus on the general case where $\mathcal{A}_1$ or $\mathcal{A}_2$ might not be smooth or simply convex. We explain how we can recover and compute the terms of (18). In the general case, Eq. (12) becomes:

$$\boldsymbol{x}^* - \boldsymbol{x}_1^*(\boldsymbol{x}^*) + T(\boldsymbol{q})\boldsymbol{x}_2^*(\boldsymbol{x}^*, \boldsymbol{q}) = 0, \tag{20}$$

with:

$$\begin{cases} \boldsymbol{x}_1^*(\boldsymbol{x}^*) \in \partial\sigma_{\mathcal{A}_1}(-\boldsymbol{x}^*), \\ \boldsymbol{x}_2^*(\boldsymbol{x}^*, \boldsymbol{q}) \in \partial\sigma_{\mathcal{A}_2}(R(\boldsymbol{q})^\top\boldsymbol{x}^*), \end{cases}$$

which are computed by the GJK+EPA procedure. By casually writing $\nabla\sigma_{\mathcal{A}_1}(-\boldsymbol{x}^*) = \boldsymbol{x}_1^*(\boldsymbol{x}^*)$ and $\nabla\sigma_{\mathcal{A}_2}(R(\boldsymbol{q})^T\boldsymbol{x}_2^*) = \boldsymbol{x}_2^*(\boldsymbol{x}^*, \boldsymbol{q})$, we recover (12) and ultimately (18).

Uninformative gradients: the case of meshes. When a shape is non-smooth or non-strictly convex, the Hessian of its support function may be null (e.g., a flat surface) or even undefined (e.g., the Hessian at the vertex of a cube). Let us consider the example of a mesh representing the convex-hull of an arbitrary object to illustrate this phenomenon. A mesh $\mathcal{A}$ is a list of N_v vertices $\{\boldsymbol{v}_1, ...\boldsymbol{v}_{N_v}\}$ and therefore:

$$\begin{aligned} \sigma_{\mathcal{A}}(\boldsymbol{x}) &= \max_{\boldsymbol{v}_i \in \{\boldsymbol{v}_1, ..., \boldsymbol{v}_{N_v}\}} \langle \boldsymbol{v}_i, \boldsymbol{x} \rangle, \\ \nabla\sigma_{\mathcal{A}}(\boldsymbol{x}) &= \boldsymbol{v}_{i^*}, \end{aligned} \tag{21}$$

for a certain $i^* \in [1, N_v]$. We define the matrix $V \in \mathbb{R}^{3\times N_v}$ and the vector $\boldsymbol{z}(\boldsymbol{x}) \in \mathbb{R}^{N_v}$ as:

$$\begin{aligned} V^T &= (\boldsymbol{v}_1 \cdots \boldsymbol{v}_{N_v})^T, \\ \boldsymbol{z}(\boldsymbol{x}) &= V^T\boldsymbol{x}. \end{aligned} \tag{22}$$

This allows us to define the vector of *argmax weights* $\boldsymbol{a}(\boldsymbol{z}) \in \mathbb{R}^{N_v}$:

$$\boldsymbol{a}(\boldsymbol{z}) = \underset{\|\boldsymbol{w}\|_1 \leq 1,\ \boldsymbol{0} \leq \boldsymbol{w}}{\arg\max}\ \boldsymbol{z}^T\boldsymbol{w}. \tag{23}$$

Therefore, all components of $\boldsymbol{a}(\boldsymbol{z}(\boldsymbol{x}))$ are null, except the $i^{*\text{th}}$ component which value is 1. As a consequence, we have:

$$\nabla\sigma_{\mathcal{A}}(\boldsymbol{x}) = \boldsymbol{v}_{i^*} = \sum_i a_i\boldsymbol{v}_i = V\boldsymbol{a}(\boldsymbol{z}(\boldsymbol{x})), \tag{24}$$

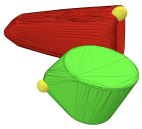
and:

$$\frac{\partial^2\sigma_{\mathcal{A}}(\boldsymbol{x})}{\partial\boldsymbol{x}^2} = V\frac{\partial\boldsymbol{a}(\boldsymbol{z})}{\partial\boldsymbol{z}}V^\top = \boldsymbol{0}, \tag{25}$$

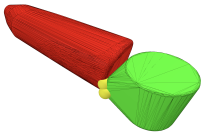
because $\partial\boldsymbol{a}(\boldsymbol{y})/\partial\boldsymbol{y}$ is null almost everywhere. As a consequence, the gradients of $\boldsymbol{x}^*$ w.r.t $\boldsymbol{q}$ obtained after solving (18) fail to capture information regarding the curvature of the underlying object which the mesh approximates.

Hessian estimation via randomized smoothing. To overcome this issue, we use a randomized smoothing approach in order to estimate $\partial^2\sigma_{\mathcal{A}}(\boldsymbol{x})/\partial\boldsymbol{x}^2$. We apply (5) to $g = \nabla\sigma_{\mathcal{A}}$ to obtain:

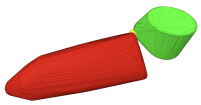
$$\frac{\partial^2\sigma_{\mathcal{A}}(\boldsymbol{x})}{\partial\boldsymbol{x}^2} \approx \frac{1}{M}\sum_{j=0}^{M} -\nabla\sigma_{\mathcal{A}}(\boldsymbol{x} + \epsilon\boldsymbol{z}^{(j)})\frac{\nabla\log\mu(\boldsymbol{z}^{(j)})^\top}{\epsilon}. \tag{26}$$



(a) Finite differences.



(b) 0^{th}-order.



(c) 1^{st}-order. Gumbel and Gaussian.

Fig. 3: **Convex hulls of YCB shapes, goal: find a relative pose such that the yellow points are contact points.** We display the final pose each method converged to. Finite differences fail to generate a satisfying pose. The 0^{th}-order gradient estimator is better than finite differences but is not as precise as the 1^{st}-order gradient estimator.

In practice, we choose μ to be a normalized Gaussian centered in $\mathbf{0}$. Although this procedure to evaluate the Hessian of the support function requires to compute the support function M times, it is in practice very efficient, as first the computation of the support function is very cheap, second it can be warm-started and third it is highly parallelisable. Finally, this method to estimate the Hessian of a support function is generic as it can be applied to *any* convex shape with a computationally tractable support function.

Special case of meshes: the Gumbel distribution. In the specific case of meshes, the structure of the support function allows to replace the Gaussian distribution by the Gumbel distribution [23], resulting in a closed form solution to estimate the mean of $\partial^2\sigma_{\mathcal{A}}(\boldsymbol{x})/\partial\boldsymbol{x}^2$ and thus removing the need of a Monte-Carlo estimator. Thus, using a Gumbel distribution μ with zero mean and identity matrix variance to sample the noise, we get a closed-form solution for (3) when applied to $g(\boldsymbol{z}) = \boldsymbol{a}(\boldsymbol{z})$ from Eq. (23):

$$\begin{aligned}\boldsymbol{a}_\epsilon(\boldsymbol{z}) &= \mathbb{E}_{Z\sim\mu}\left[a(\boldsymbol{z}+\epsilon Z)\right] \\ &= \frac{1}{\sum_j e^{z_j/\epsilon}}\left(e^{z_1/\epsilon} \;\dots\; e^{z_{N_v}/\epsilon}\right)^\top. \end{aligned} \tag{27}$$

The smoothed $\boldsymbol{a}_\epsilon$ is thus simply a soft-max, which is a smooth and differentiable function of $\boldsymbol{y}$ [8]. The ϵ parameter serves as a temperature parameter [23]. Due to the nature of the soft-max operator, the i^{th} weight in $\boldsymbol{a}_\epsilon$ decreases exponentially the further $z_i = \langle \boldsymbol{v}_i, -\boldsymbol{x}^*\rangle$ is from the maximum value $\sigma_{\mathcal{A}}(-\boldsymbol{x}^*)$. This maximum value is attained by the witness point of the considered shape $\mathcal{A}$. We illustrate in Fig. 2 the weighting of this soft-max operation on a mesh. Remarkably, the further a vertex is from the current witness point, the less it contributes to the soft-max. It is therefore only necessary to keep the points of the mesh which belong to a neighborhood around the witness point of shape $\mathcal{A}$. This fact renders the use of the Gumbel distribution very efficient on meshes. By choosing the *depth* of neighboring vertices around the witness point, which we denote by n_l, we can choose to limit or increase the number of neighbors involved in the computation of (27). As an example, a depth of $n_l = 2$ corresponds to keeping only the neighbors of the witness points and the neighbors of neighbors.

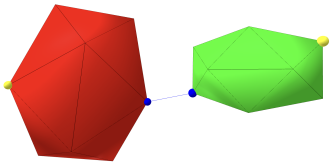


Fig. 4: **Rough shapes, goal: find a relative pose such that the yellow points are contact points.** The current witness-points are the blue points.

Finally, by applying the chain rule we get the estimation of $\partial^2\sigma_{\mathcal{A}}(\boldsymbol{x})/\partial\boldsymbol{x}^2$:

$$\frac{\partial^2\sigma_{\mathcal{A}}(\boldsymbol{x})}{\partial\boldsymbol{x}^2} \approx \frac{\partial\boldsymbol{a}_\epsilon(\boldsymbol{z}(\boldsymbol{x}))}{\partial\boldsymbol{x}} = V\frac{\partial\boldsymbol{a}_\epsilon(\boldsymbol{z})}{\partial\boldsymbol{z}}V^T, \tag{28}$$

where $\partial\boldsymbol{a}_\epsilon(\boldsymbol{z})/\partial\boldsymbol{z}$ is simply the derivative of the soft-max function.

Derivatives of the witness points. In general, $\boldsymbol{x}_1^* = \nabla\sigma_{\mathcal{A}_1}(-\boldsymbol{x}^*)$ so by applying the chain rule we get:

$$\frac{\partial\boldsymbol{x}_{1,2}^*}{\partial\boldsymbol{q}} = -\frac{\partial\nabla\sigma_{\mathcal{A}_{1,2}}(-\boldsymbol{x}^*)}{\partial\boldsymbol{x}^*}\frac{\partial\boldsymbol{x}^*}{\partial\boldsymbol{q}} = -\frac{\partial^2\sigma_{\mathcal{A}_{1,2}}(-\boldsymbol{x}^*)}{\partial\boldsymbol{x}^{*2}}\frac{\partial\boldsymbol{x}^*}{\partial\boldsymbol{q}}. \tag{29}$$

To conclude this section, we have introduced a complete approach to retrieve a 1^{st}-order estimate of the variation of the witness points lying on the two shapes, according to the relative placements between these two. In particular, in the case of meshes, we have proposed a closed-form formula to compute a local approximation of the Hessian using the neighborhood of the current witness points.

V. Experiments

In this section, we evaluate whether the gradients obtained with the two proposed estimators are meaningful and how computationally efficient it is to compute them compared to finite differences.

Contact-pose generation benchmark. To answer the first question, we evaluate the 0^{th}-order and 1^{st}-order estimators against finite differences on a synthetic benchmark of non-trivial minimization problems. We generate 100 collision pairs with random polyhedral ellipsoids, i.e., ellipsoids which surfaces are represented by a 12-vertices convex mesh (see Fig. 4). The resulting shapes are rough, i.e., the curvature information of the original shape has been greatly truncated. For each pair of convex shapes $(\mathcal{A}_1, \mathcal{A}_2)$, we generate 100 random target points $\boldsymbol{x}_{1,\text{des}} \in \mathcal{A}_1$ and $\boldsymbol{x}_{2,\text{des}} \in \mathcal{A}_2$ on the shapes' surfaces. For each of the 10000 generated problems, the goal is to find a relative pose $T(\boldsymbol{q})$ between $\mathcal{A}_1$ and $\mathcal{A}_2$ such that the shapes are in contact and their witness points $\boldsymbol{x}_1^*(\boldsymbol{q})$ and $\boldsymbol{x}_2^*(\boldsymbol{q})$ satisfy $\boldsymbol{x}_1^*(\boldsymbol{q}) = \boldsymbol{x}_{1,\text{des}}$ and $\boldsymbol{x}_2^*(\boldsymbol{q}) = \boldsymbol{x}_{2,\text{des}}$. Mathematically, this corresponds to solving the minimization problem:

$$\min_{\boldsymbol{q}} \frac{1}{2}\sum_{i=1,2}\left\|\boldsymbol{x}_i^*(\boldsymbol{q}) - \boldsymbol{x}_{i,\text{des}}^*\right\|^2 + \frac{1}{2}\left\|\boldsymbol{x}_1^*(\boldsymbol{q}) - \boldsymbol{x}_2^*(\boldsymbol{q})\right\|^2. \tag{30}$$

	Finite differences		0^{th}-order Gaussian	1^{st}-order Gaussian	1^{st}-order Gumbel
	$M = 12$ $\epsilon = 10^{-6}$	$M = 12$ $\epsilon = 10^{-3}$	$M = 50$ $\epsilon = 10^{-2}$	$M = 20$ $\epsilon = 10^{-3}$	$n_l = 1$ $\epsilon = 10^{-4}$
D1	2×10^{-33}	8×10^{-33}	4×10^{-23}	4×10^{-16}	6×10^{-16}
Q1	4×10^{-32}	7×10^{-23}	5×10^{-20}	1×10^{-10}	3×10^{-10}
Median	3×10^{-3}	4×10^{-14}	2×10^{-13}	4×10^{-8}	1×10^{-8}
Q3	4×10^{-2}	1×10^{-2}	2×10^{-3}	3×10^{-7}	7×10^{-8}
D9	8×10^{-2}	5×10^{-2}	5×10^{-3}	2×10^{-6}	2×10^{-5}

TABLE I: **Rough shapes (see Fig. 4), value of $C(\boldsymbol{q})$ after 50 iterations of Gauss-Newton with line search.** Q3 and D9: respectively 25% and 10% of problems have a terminal cost worse (higher) than the reported value.

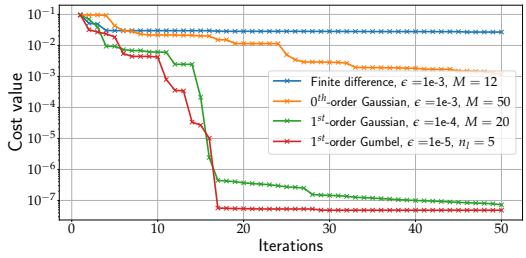


Fig. 5: **YCB shape, goal: find a relative pose such that the yellow points are contact points.** Finite difference typically gets stuck around a limited precision (depending on the finite difference increment). The 1^{st}-order estimator converges rapidly towards a solution with high precision unlike the 0^{th}-order estimator.

To solve this minimization problem, we use the Gauss-Newton algorithm with backtracking line search [24] and run it for 50 iterations. To compute the Jacobian of the cost, we evaluate the terms $\partial \boldsymbol{x}^*_{1,2}/\partial \boldsymbol{q}$, with finite differences, and the 0^{th} and 1^{st}-order estimators proposed in this work. Finally, to compute $\boldsymbol{x}^*_1(\boldsymbol{q})$ and $\boldsymbol{x}^*_2(\boldsymbol{q})$, we use the combination of the GJK and EPA algorithms implemented in the HPP-FCL library [25], [26], a fork of the FCL library [27].

We report as quantiles in Table I the value of the terminal cost $C(\boldsymbol{q})$. The lower this quantity, the better the quality of the solution found. We are particularly interested in the quantiles Q3 and D9 of Table I: respectively 25% and 10% of problems have a terminal cost worse (higher) than the reported value. The higher this value is, the less reliable the method is. We observe that finite differences have a high value for Q3 and D9 whereas the two proposed estimators are at least one order of magnitude better. Remarkably, the 1^{st}-order estimators, whether the underlying distribution used is Gaussian or Gumbel, is extremely accurate and reliable, with a value of Q3 and D9 at least 3 orders of magnitude better than finite differences.

As an additional qualitative example, Fig. 3 and Fig. 5 show a typical example of failure of finite differences on a collision pair of the YCB dataset - a dataset which contains high-resolution meshes of real-world household objects [28]. Finally, Fig. 6 shows the typical impact of the noise and number of samples on the 1^{st}-order estimator using the Gumbel distribution for the same YCB problem. The same kind of behavior is obtained when using a Gaussian distribution. Overall, the higher the number of samples, the better the quality of the estimator; the higher the noise, the faster the convergence at the price of reduced accuracy.

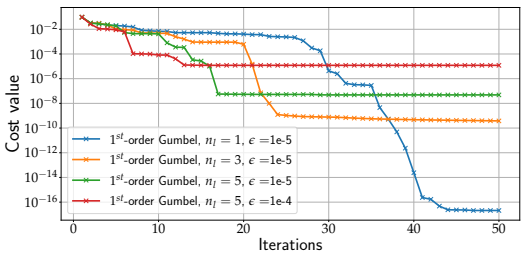


Fig. 6: **Goal: find a relative pose such that the yellow points are contact points.** With more noise, the faster the convergence is at first, but the estimator ends up being less precise, hence converges to a less optimal solution.

Method	Parameters	Timings (in μs) Collision	No collision
Finite differences	$M = 12$	78 ± 48	11.0 ± 7.3
0^{th}-order Gaussian	$M = 10$	65 ± 36	9 ± 6
	$M = 20$	122 ± 130	17 ± 10
	$M = 50$	338 ± 391	47 ± 30
	$M = 100$	613 ± 402	86 ± 45
1^{st}-order Gaussian	$M = 10$	4.8 ± 1.4	3.1 ± 0.3
	$M = 20$	8.9 ± 2.8	5.8 ± 0.5
	$M = 50$	22 ± 8	15 ± 3
	$M = 100$	42 ± 13	27 ± 3
1^{st}-order Gumbel	$n_l = 1$	1.7 ± 0.5	1.6 ± 0.5
	$n_l = 3$	4.1 ± 1.8	3.9 ± 1.4
	$n_l = 5$	9.6 ± 7.8	9.7 ± 8.1

TABLE II: **Timings for computing collision detection derivatives, for collision pairs of the YCB dataset.** The parameters were selected in the ranges which would typically be used in practice.

YCB timings benchmarks. To evaluate the computational efficiency of the proposed estimators, we generate 10000 collision detection problems (1) using meshes from the YCB real-world objects dataset [28] and measure the time taken to compute the derivatives of witness points for each estimator. For the finite differences and 0^{th}-order, GJK+EPA are warm started at each sample $\boldsymbol{q} + \epsilon \boldsymbol{z}$ to enhance the computational efficiency. The results, reported in Table II show that although the 0^{th}-order estimator is often prohibitive compared to finite differences, the 1^{st}-order estimators using Gaussian and Gumbel distributions can be obtained extremely efficiently, on the order of the micro-seconds and from 10 to 70 times faster than finite differences.

VI. Conclusion

In this paper, we propose a generic approach for computing 0^{th} and $1^{\text{th}}-$order derivatives of collision detection for *any* convex shapes by leveraging randomized smoothing techniques. While being robust and easy to implement, our approach exhibits strong benefits in terms of speed and accuracy, taking only few micro-seconds to compute informative derivatives of complex shapes, such as meshes with hundred vertices, involved in real robotic applications. Remarkably, the $1^{\text{th}}-$order estimators are also both more accurate and less expensive to compute than the 0^{th}-order estimator. All these gradient estimation methods have been implemented in the HPP-FCL and Pinocchio ecosystems. We plan to extend our contributions by applying these methods for differentiable simulation, optimal grasp synthesis and trajectory optimization.

ACKNOWLEDGMENTS

This work was partly supported by the European Regional Development Fund under the project IMPACT (reg. no. CZ.02.1.01/0.0/0.0/15 003/0000468), by the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute) and the Louis Vuitton ENS Chair on Artificial Intelligence.

REFERENCES

[1] C. Ericson, *Real-Time Collision Detection*. The Morgan Kaufmann Series, 2004.

[2] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter, "End-to-end differentiable physics for learning and control," *Advances in neural information processing systems*, vol. 31, 2018.

[3] J. Carpentier and N. Mansard, "Analytical derivatives of rigid body dynamics algorithms," in *Robotics: Science and systems (RSS 2018)*, 2018.

[4] M. Toussaint, K. Allen, K. Smith, and J. Tenenbaum, "Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning," in *Robotics: Science and Systems XIV*, Robotics: Science and Systems Foundation, June 2018.

[5] Q. Le Lidec, I. Kalevatykh, I. Laptev, C. Schmid, and J. Carpentier, "Differentiable simulation for physical system identification," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3413–3420, 2021.

[6] M. Geilinger, D. Hahn, J. Zehnder, M. Bächer, B. Thomaszewski, and S. Coros, "ADD: Analytically Differentiable Dynamics for Multi-Body Systems with Frictional Contact," *arXiv:2007.00987 [cs]*, July 2020.

[7] K. Werling, D. Omens, J. Lee, I. Exarchos, and C. K. Liu, "Fast and Feature-Complete Differentiable Physics for Articulated Rigid Bodies with Contact," *arXiv:2103.16021 [cs, eess]*, June 2021.

[8] Q. Berthet, M. Blondel, O. Teboul, M. Cuturi, J.-P. Vert, and F. Bach, "Learning with differentiable pertubed optimizers," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, eds.), vol. 33, pp. 9508–9519, Curran Associates, Inc., 2020.

[9] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE International Symposium on System Integrations (SII)*, 2019.

[10] E. Gilbert, D. Johnson, and S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, 1988.

[11] G. Van den Bergen, "Proximity Queries and Penetration Depth Computation on 3D Game Objects," *In Game Developers Conference*, 2001.

[12] L. Montaut, Q. Lidec, V. Petrík, J. Sivic, and J. Carpentier, "Collision Detection Accelerated: An Optimization Perspective," in *Proceedings of Robotics: Science and Systems*, (New York City, NY, USA), June 2022.

[13] K. Mamou and F. Ghorbel, "A Simple and Efficient Approach for 3D Mesh Approximate Convex Decomposition," in *The 16th IEEE International Conference on Image Processing*, 2009.

[14] G. Snethen, "Xenocollide: Complex collision made simple," in *Game Programmings Gems*, 2008.

[15] J. Solà, J. Deray, and D. Atchuthan, "A micro Lie theory for state estimation in robotics," 2021.

[16] J. Matyas *et al.*, "Random optimization," *Automation and Remote control*, vol. 26, no. 2, pp. 246–253, 1965.

[17] B. Polyak, *Introduction to Optimization*. 07 1987.

[18] J. C. Duchi, P. L. Bartlett, and M. J. Wainwright, "Randomized smoothing for stochastic optimization," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 674–701, 2012.

[19] Q. Le Lidec, I. Laptev, C. Schmid, and J. Carpentier, "Differentiable rendering with perturbed optimizers," *Advances in Neural Information Processing Systems*, vol. 34, pp. 20398–20409, 2021.

[20] F. Petersen, B. Goldluecke, C. Borgelt, and O. Deussen, "Gendr: A generalized differentiable renderer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4002–4011, 2022.

[21] Q. L. Lidec, L. Montaut, C. Schmid, I. Laptev, and J. Carpentier, "Leveraging randomized smoothing for optimal control of nonsmooth dynamical systems," *arXiv preprint arXiv:2203.03986*, 2022.

[22] H. J. T. Suh, T. Pang, and R. Tedrake, "Bundled gradients through contact via randomized smoothing," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4000–4007, 2022.

[23] E. J. Gumbel, *Statistical theory of extreme values and some practical applications: a series of lectures*, vol. 33. US Government Printing Office, 1954.

[24] S. Wright, J. Nocedal, *et al.*, "Numerical optimization," *Springer Science*, vol. 35, no. 67-68, p. 7, 1999.

[25] J. Mirabel, S. Tonneau, P. Fernbach, A.-K. Seppälä, M. Campana, N. Mansard, and F. Lamiraux, "HPP: A new software for constrained motion planning," in *International Conference on Intelligent Robots and Systems*, 2016.

[26] J. Pan, S. Chitta, D. Manocha, F. Lamiraux, J. Mirabel, J. Carpentier, *et al.*, "HPP-FCL: an extension of the Flexible Collision Library." https://github.com/humanoid-path-planner/hpp-fcl, 2015–2022.

[27] J. Pan, S. Chitta, and D. Manocha, "FCL: A General Purpose Library for Collision and Proximity Queries," in *2012 IEEE International Conference on Robotics and Automation*, IEEE, 2012.

[28] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The ycb object and model set: Towards common benchmarks for manipulation research," in *2015 international conference on advanced robotics (ICAR)*, pp. 510–517, IEEE, 2015.