PONG: Probabilistic Object Normals for Grasping via Analytic Bounds on Force Closure Probability

Albert H. Li[†], Preston Culbertson[‡], Aaron D. Ames[†,‡]

***Abstract*— Classical approaches to grasp planning are deterministic, requiring perfect knowledge of an object's pose and geometry. In response, data-driven approaches have emerged that plan grasps entirely from sensory data. While these data-driven methods have excelled in generating parallel-jaw and power grasps, their application to precision grasps (those using the fingertips of a dexterous hand, e.g, for tool use) remains limited. Precision grasping poses a unique challenge due to its sensitivity to object geometry, which allows small uncertainties in the object's shape and pose to cause an otherwise robust grasp to fail. In response to these challenges, we introduce Probabilistic Object Normals for Grasping (PONG), a novel, analytic approach for calculating a conservative estimate of force closure probability in the case when contact locations are known but surface normals are uncertain. We then present a practical application where we use PONG as a grasp metric for generating robust grasps both in simulation and real-world hardware experiments. Our results demonstrate that maximizing PONG efficiently produces robust grasps, even for challenging object geometries, and that it can serve as a well-calibrated, uncertainty-aware metric of grasp quality.**

I. Introduction

Grasp synthesis has been a canonical problem in robotic manipulation since the field's inception. Despite decades of work, dexterous grasps are still challenging to synthesize, since multifinger hands have complex kinematics and high-dimensional grasp parameterizations. Broadly speaking, two types of approaches toward dexterous grasp synthesis exist. *Analytic* approaches evaluate grasp quality using a *metric* [1] and then maximize it using any optimization technique. Though analytic methods offer principled guarantees, they suffer from two problems: (i) they usually assume perfect knowledge of an object's geometry, and (ii) they are typically hard to optimize efficiently while enforcing kinematic and collision constraints [2]. In response, many *learning-based* methods account for uncertainty with data but lack guarantees, often checking constraints are satisfied *post hoc* via rejection sampling rather than enforcing them during synthesis [3], [4], [5]. To ensure sample efficiency, large amounts of high-quality data are required, which may be difficult to generate or collect.

Ideally, a grasp synthesis method should be uncertainty-aware, computationally-efficient, and generalize to a broad class of object geometries. To that end, our contributions are as follows. First, we develop a novel analytic theory of probabilistic force closure, PONG: **P**robabilistic **O**bject **N**ormals for **G**rasping. Given known contact locations and a model of surface normal uncertainty, PONG computes a lower bound on the probability that a grasp is force closure, which we maximize to synthesize uncertainty-aware precision grasps. We demonstrate PONG's effectiveness by using it as a curvature-aware grasp metric in both simulation[1] and hardware experiments, where we treat the object's curvature as a proxy for uncertainty about its surface geometry. We show in simulation that, as PONG increases, the failure rate in grasps decreases dramatically. In hardware, even under challenging real-world conditions and using approximate, learning-based object models constructed only from visual data, we can achieve a nearly 70% grasp success rate.

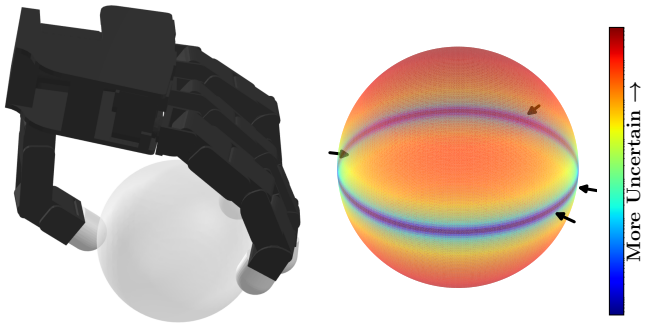


Fig. 1: PONG synthesizes *uncertainty-aware* dexterous precision grasps by extending the concept of *force closure* to the probabilistic setting. In this example, we artificially impose uncertainty in the target object's surface normals, and PONG computes a locally-optimal grasp with fingertips near the equator, the region with the least uncertainty.

A. Related Work

While this paper presents a novel analytic treatment of the *probability of force closure* (PFC), prior work has studied PFC from data-driven and empirical perspectives. For example, [6] notes that the traditional Ferrari-Canny epsilon-ball metric is sensitive to uncertainty in the object pose; they generate a Monte Carlo estimate of PFC via simulation, and use this to rank grasps by robustness. *Gaussian process implicit surfaces* (GPISs) [7] have also been an influential way to represent the object geometry itself as uncertain, and have been deployed successfully for parallel-jaw grasping [8], dexterous grasping with tactile sensors [9], [10], and for control synthesis [11]. While GPISs can reason jointly over both uncertain contact points and normals, they have several drawbacks, including being difficult to supervise from sensor data (requiring ground-truth signed-distance labels),

† A. H. Li and A. D. Ames are with the Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA 91125, USA, {alberthli, ames}@caltech.edu.

‡ P. Culbertson and A. D. Ames are with the Department of Civil and Mechanical Engineering, California Institute of Technology, Pasadena, CA 91125, USA, {pculbert, ames}@caltech.edu.

[1]Open-source implementation available at github.com/alberthli/pong.

poor scaling with respect to amount of data, and lack of expressivity (due to the strong smoothness prior imposed on the object surface by typical choices of the kernel function).

An alternate approach is to instead directly learn probabilistic grasp metrics from data. For example, Dex-Net 2.0 predicts the robustness of a batch of planar parallel-jaw grasps, then selects the most robust one [12]. A similar idea has been applied to multifinger hands in a gradient-based optimization setting with joint limits by leveraging the differentiability of neural networks [13], [14].

This paper is most similar in spirit to recent work on differentiable approximations of analytic metrics for fast gradient-based grasp synthesis. Proposed methods include solving sequences of SDPs [15]; solving a sum of squares program [16]; optimizing a differentiable relaxation of the force closure condition [17], [18]; solving a bilinear optimization program with a QP force closure constraint [19]; and maximizing an almost-everywhere differentiable proxy for the Ferrari-Canny metric in a bilevel setting [20]. The common theme of these works is the use of nonlinear optimization to enforce (or penalize) kinematic, collision, and contact constraints. Our metric, PONG, is computed by solving LPs, allowing it to serve as an objective for similar bilevel programming approaches to grasp synthesis.

B. Preliminaries

We consider grasp planning for a fixed-base, fully-actuated rigid-body serial manipulator with a multi-finger hand. Denote the robot configuration $q \in \mathcal{Q}$. Assume the hand has n_f fingers contacting the object at points $\{x^i\}_{i=1}^{n_f}$ with inward pointing surface normals $\{n^i\}_{i=1}^{n_f}$. Denote the forward kinematics maps $x^i = FK^i(q)$ with corresponding (translational) Jacobians $J^i(q) \in \mathbb{R}^{3\times n}$. Denote the single rigid object $\mathcal{O}$ with surface $\partial\mathcal{O}$.

As shorthand, let $\mathcal{C}_x := \text{conv}(\{x_l\}_l)$ denote the convex hull of a finite set of points indexed by l. Define the *wedge* operator $[\cdot]_\times : \mathbb{R}^3 \to \mathfrak{so}(3)$ such that $[a]_\times b := a \times b$ for $a, b \in \mathbb{R}^3$. We parameterize grasps with an optimized *feasible* configuration q^*, which means no undesired collisions while contacting the object. We model the fingers as point contacts with Coulomb friction.

Recall that under the Coulomb friction model, a contact force f satisfies the *no-slip* condition if for a fixed friction coefficient μ, $\|f_t\| \le \mu \cdot f_n$, where t and n denote tangent and (positive) normal components of force f respectively. We call such forces *Coulomb-compliant*. A force applied at point x induces the torque $\tau = x \times f$, which in turn induces a corresponding wrench $w = (f, \tau)$.

The *friction cone* at a point x is the cone centered around the surface normal n consisting of all Coulomb-compliant forces that can be applied. It is common to consider a pyramidal approximation of this cone [21] with n_s sides. We call its edges *basis forces* and the induced wrenches *basis wrenches*. We emphasize that the basis wrenches depend on the basis forces which themselves depend on the surface normal at a point x, a relation we use heavily below.

Let $n_w = n_c n_s$ denote the number of basis wrenches (indexed w_j^i), each associated with a finger i and pyramid edge j. For clarity, we sometimes combine the indices i, j into a single one l. Further, let $\mathcal{I} = \{1, \dots, n_f\}$, $\mathcal{J} = \{1, \dots, n_s\}$, and $\mathcal{L} = \{1, \dots, n_w\}$ denote finger, pyramid edge, and basis wrench index sets respectively.

II. A Probabilistic Notion of Force Closure

This paper extends the classical theory of *force closure* to the probabilistic setting in which the surface normals n^i are random variables. To review, we say a grasp is (deterministically) *force closure* if for any disturbance wrench it can generate Coulomb-compliant forces on the object to resist the disturbance. A well-known sufficient condition for force closure is that the origin is contained in the convex hull of a grasp's basis wrenches, i.e., $0 \in \mathcal{C}_w$ [22]. We say that such basis wrenches *certify* force closure.

In the probabilistic setting, the normals n^i are random variables, so the basis forces and wrenches are as well. This begs the central question of the paper: given known contact locations and randomly-distributed surface normals, what is (a bound on) the probability that the induced basis wrenches certify force closure?

Let the known contacts be denoted $\mathcal{X} = \{x^1, \dots, x^{n_f}\}$ and consider the set of normals that induce basis wrenches certifying force closure. We call this the *force closure set*:

$$\mathfrak{N}(\mathcal{X}) := \left\{ \{n^i\}_{i=1}^{n_f} \mid 0 \in \mathcal{C}_w \right\}. \tag{1}$$

Suppose the normals are jointly distributed with some density function $p\left(n^1, \dots, n^{n_f}\right)$ and that they are mutually independent such that p can be factorized as $\prod_{i=1}^{n_f} p(n^i)$. Then, the probability of force closure can be computed by the integral

$$P_{\text{fc}} = \mathbb{P}[0 \in \mathcal{C}_w] = \int_{\mathfrak{N}(\mathcal{X})} \prod_{i=1}^{n_f} p(n^i) dn^i. \tag{2}$$

While succinct, (2) is not directly useful for two reasons. First, integrating the density p over $\mathfrak{N}$ is difficult, because even though p is factorizable, the integral itself is not. The integration variables are coupled by the domain $\mathfrak{N}$, since $\mathcal{C}_w$ depends on all of the random normals. Second, $\mathfrak{N}$ is difficult to parameterize; it has no closed form since it is implicitly defined by the convex hull condition (1), where hull membership is checked by solving an LP [22].

Thus, our strategy is to derive an *approximate force closure set* $\mathcal{A} \subseteq \mathfrak{N}$ in tandem with a choice of density function p for which the integral over $\mathcal{A}$ is known. Since $\mathcal{A} \subseteq \mathfrak{N}$, integrating p over $\mathcal{A}$ yields a lower bound on P_{fc}.

The following result [23] will motivate our constructions by providing an exact method for integrating a bivariate Gaussian density over an arbitrary planar polygon.

Proposition 1. *Let $\mathscr{P}$ be a planar polygon with M vertices $y^1, \dots, y^M \in \mathbb{R}^2$ ordered counterclockwise with $y^{M+1} := y^1$. Let Z be a bivariate Gaussian random variable with mean μ and covariance $\Sigma = \text{diag}(\sigma_1^2, \sigma_2^2)$. Then,*

$$\mathbb{P}[Z \in \mathscr{P}] = \frac{1}{\sigma_2\sqrt{8\pi}} \sum_{m=1}^{M} D^m \int_0^1 A^m(r) B^m(r) dr, \tag{3}$$

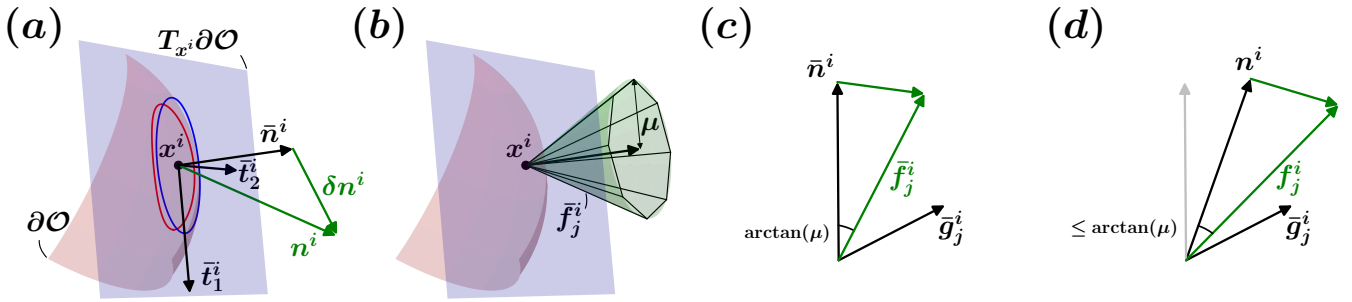


Fig. 2: **(a)** The parameterization from Sec. III-A. For visual clarity, we point the contact normals outward. The variances defining the uncertainty ellipse (blue) lie in the tangent plane at point x^i on surface $\partial\mathcal{O}$. A perturbation δn^i is drawn to generate a random normal n^i. **(b)** The friction cone (green) and a canonical pyramidal approximation. Each basis wrench $\bar{w}_j^i$ depends on position x^i, surface normal $\bar{n}^i$, and friction coefficient μ. **(c)** The construction of a mean basis force $\bar{f}_j^i$ using generator $\bar{g}_j^i$. **(d)** The construction of a random basis force f_j^i using the same generator as in the mean case. Note the smaller angle, which ensures the random basis forces are Coulomb-compliant.

where the terms above are

$$D^m := y_2^{m+1} - y_2^m,$$
$$A^m(r) := \exp\left(-\frac{1}{2\sigma_2^2}\left[(1-r)y_2^m + r y_2^{m+1} - \mu_2\right]^2\right),$$
$$B^m(r) := \operatorname{erf}\left(\frac{(1-r)y_1^m + r y_1^{m+1} - \mu_1}{\sigma_1\sqrt{2}}\right).$$

This allows us to numerically integrate over planar regions with just line integrals (full proof in A).

III. PONG: A Tractable PFC Lower Bound

This section presents PONG, which leverages Proposition 1 to address the aforementioned challenges. Due to space constraints, we elide computational details to our open-source implementation and the Appendix.

We proceed in three steps: (A) we present a construction that linearly relates a random surface normal n^i to a set of random basis wrenches w_j^i; (B) we use this relation to define disjoint sets $\mathcal{A}_i$ such that $\mathcal{A} = \bigcup_i \mathcal{A}_i$, which allows us to factorize the integral of p over $\mathcal{A}$; and (C) we parameterize the sets $\mathcal{A}_i$ in an integrable way and compute them with linear programming. The end result is the lower bound

$$\begin{aligned} L_{\text{fc}} := \prod_{i=1}^{n_f} \int_{\mathcal{A}^i} p(n^i)dn^i &= \int_{\mathcal{A}} \prod_{i=1}^{n_f} p(n^i)dn^i \\ &\leq \int_{\mathfrak{N}} \prod_{i=1}^{n_f} p(n^i)dn^i, \end{aligned} \tag{4}$$

which is the main result of the paper.

A. Random Normals, Forces, and Wrenches

We begin with our representation of the random surface normals n^i. We model each random normal as the sum of a deterministic *mean normal vector* denoted $\bar{n}^i$ and a *random perturbation vector* $\delta n^i = \mathcal{T}_i\epsilon$, where $\mathcal{T}_i = \begin{bmatrix}\bar{t}_1^i & \bar{t}_2^i\end{bmatrix} \in \mathbb{R}^{3\times 2}$ is a basis for the tangent plane at $\bar{n}^i$, and $\epsilon \sim \mathcal{N}(0, \Sigma^i)$ is a zero-mean Gaussian random vector in $\mathbb{R}^2$ (see Fig. 2a).

We opt to parameterize the uncertain normals in this way for two reasons. First, all perturbations in the direction of $\bar{n}^i$ do not change the direction of the mean normal, and therefore do not represent any uncertainty in its orientation. Second, the planar restriction will allow us to invoke the tractable Gaussian density integral given in Proposition 1.

We remark that with this construction, the random normals n^i will not have unit norm. However, since friction cones are invariant under scaling, as long as the random basis forces f_j^i remain Coulomb-compliant, we can still certify force closure with the induced random basis wrenches.

Leveraging this insight, we now introduce a procedure for constructing a random friction pyramid about the random normal n^i such that its edges are always Coulomb-compliant. First, consider the *mean* basis wrenches $\bar{w}_j^i$ with force and torque components $\bar{f}_j^i$ and $\bar{\tau}_j^i$. Per Fig. 2b, we represent $\bar{f}_j^i$ as the sum of $\bar{n}^i$ and a tangent component $\left(f_j^i\right)_t$ of length μ. We compute $\left(f_j^i\right)_t$ using a unit length *generator* $\bar{g}_j^i(\bar{n}^i) \in \mathbb{R}^3$ (see Fig. 2c) of our choice orthogonal to $\bar{n}^i$ such that

$$\left(f_j^i\right)_t = \mu\left(\bar{g}_j^i \times \bar{n}^i\right). \tag{5}$$

For example, for some arbitrary $v \neq \bar{n}^i$, we could pick $\bar{g}_j^i(\bar{n}^i) = (v \times \bar{n}^i)/\left\|v \times \bar{n}^i\right\|_2$. Thus, we can write

$$\begin{aligned} \bar{f}_j^i &= \bar{n}^i + \mu\left(\bar{g}_j^i \times \bar{n}^i\right) \\ \implies \bar{w}_j^i &= \underbrace{\begin{bmatrix} \left(I + \mu\left[\bar{g}_j^i\right]_\times\right) \\ \left[x^i\right]_\times\left(I + \mu\left[\bar{g}_j^i\right]_\times\right) \end{bmatrix}}_{:=T_j^i(\bar{n}^i)} \bar{n}^i. \end{aligned} \tag{6}$$

Generally, $\bar{w}_j^i$ is nonlinear with respect to $\bar{n}^i$ since the generators can depend nonlinearly on $\bar{n}^i$. However, if we choose to generate the *random* basis wrenches w_j^i with the generators from the mean case, we have the linear relation $w_j^i = T_j^i(\bar{n}^i)n^i$ with respect to random normal n^i.

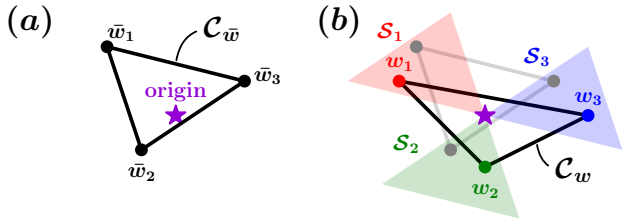
Our construction satisfies $\left\|(f_j^i)_t\right\|_2 \leq \mu\left\|n^i\right\|$ (in contrast, observe that $\left\|\bar{f}_j^i\right\|_2 = \mu\left\|\bar{n}^i\right\|$ as in Fig. 2c and 2d). To see this, let ψ_j^i be the angle between $\bar{g}_j^i$ and n^i. Then,

$$\left\|(f_j^i)_t\right\|_2 = \mu\left\|\bar{g}_j^i\right\|\left\|n^i\right\|\sin(\psi_j^i) \leq \mu\left\|n^i\right\|, \tag{7}$$

ensuring Coulomb-compliance of the random basis forces.

B. Deriving a Decomposable Approximate Force Closure Set

Next, we provide a procedure to generate disjoint sets $\mathcal{A}_i$ whose union defines the inner approximation $\mathcal{A}$ of the



The expected convex hull. A sampled convex hull.

Fig. 3: Prop. 2 can be interpreted with the sets $\mathcal{S}_l := -\mathcal{C}_{\bar{w}} + \bar{w}_l$ (colored) centered about the expected wrenches. If the random wrenches satisfy $w_l \in \mathcal{S}_l,\ \forall l \in \mathcal{L}$, then $0 \in \mathcal{C}_w$, so the sampled wrenches certify force closure.

ideal integration domain $\mathfrak{N}$, allowing us to separate the P_{fc} integral (2). We will use the following result, which certifies the random basis wrenches w_j^i form a force closure grasp.

Proposition 2. *If $w_l - \bar{w}_l \in -\mathcal{C}_{\bar{w}}$ for all $l \in \mathcal{L}$, then $0 \in \mathcal{C}_w$.*

Proof. Suppose for the sake of contradiction that $0 \notin \mathcal{C}_w$. Then, there exists a such that $\langle a, w\rangle > 0$ for all $w \in \mathcal{C}_w$ by the separating hyperplane theorem. Further, there must exist some l^* satisfying $\langle a, \bar{w}_{l^*}\rangle \leq \langle a, \bar{w}_l\rangle$ for all $\bar{w}_l \in \mathcal{C}_{\bar{w}}$. Since $w_l - \bar{w}_l \in -\mathcal{C}_{\bar{w}}$ for each l, we have corresponding to l^* a set of convex weights $\alpha^{l^*} \in \mathbb{R}^{n_w}$ such that

$$\begin{aligned} w_{l^*} &= \bar{w}_{l^*} + (w_{l^*} - \bar{w}_{l^*}) = \bar{w}_{l^*} - \sum_{l=1}^{n_w} \alpha_l^{l^*}\, \bar{w}_l \\ \implies \langle a, w_{l^*}\rangle &= \langle a, \bar{w}_{l^*}\rangle - \sum_{l=1}^{n_w} \alpha_l^{l^*}\, \langle a, \bar{w}_l\rangle \qquad (8) \\ &\leq \langle a, \bar{w}_{l^*}\rangle - \langle a, \bar{w}_{l^*}\rangle \sum_{l=1}^{n_w} \alpha_l^{l^*} = 0. \end{aligned}$$

But, $\langle a, w_{l^*}\rangle > 0$ since $w_{l^*} \in \mathcal{C}_w$, contradicting (8). □

We view $w_l - \bar{w}_l$ as the deviation of the l^{th} basis wrench from some mean value (e.g., from a model). If all deviations are contained in $-\mathcal{C}_{\bar{w}}$, then by Proposition 2, the origin lies in the convex hull $\mathcal{C}_w$ of the true, random basis wrenches, i.e., *the grasp is force closure*. See Fig. 3 for visual intuition.

Combining this with the relation $w_j^i = T_j^i n^i$ from Sec. III-A, we thus want our approximations to satisfy

$$\mathcal{A}_i \subseteq \{n^i \mid T_j^i(n^i - \bar{n}^i) \in -C_{\bar{w}},\ \forall j \in \mathcal{J}\}. \qquad (9)$$

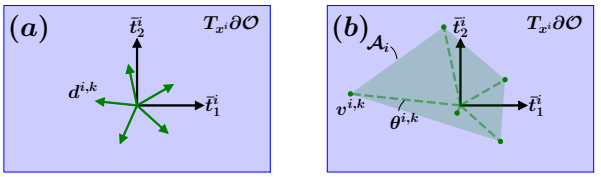
By applying Proposition 2, we have

$$n^i \in \mathcal{A}^i,\ \forall i \in \mathcal{I} \implies (n^1, \ldots, n^{n_f}) \in \mathfrak{N}, \qquad (10)$$

so letting $\mathcal{A} = \bigcup_{i=1}^{n_f} \mathcal{A}_i$, we recover the bound (4). The decomposition is possible because even though in (9), $\mathcal{C}_{\bar{w}}$ depends on all the *mean* surface normals $(\bar{n}^1, \ldots, \bar{n}^{n_f})$, these are fixed parameters of the known uncertainty distributions. In contrast, in (1), $\mathcal{C}_w$ (and thus $\mathfrak{N}$) depends jointly on all of the *random* normals, which are the integration variables.

C. Computing Approximate Force Closure Sets

Unfortunately, the condition on the right hand side of (9) cannot be expressed in closed form. Thus, we let $\mathcal{A}_i$ be a conservative polygonal approximation of it by performing a search procedure to find a polytope with large volume satisfying (9) parameterized by its vertices.



Fixed search directions $d^{i,k}$. Points $v^{i,k}$ and convex hull $\mathcal{A}_i$.

Fig. 4: Visualization of the parameterization and construction of $\mathcal{A}_i$. **(a)** Example search directions represented in the planar coordinates of the tangent basis. **(b)** Example points placed along the search directions and the corresponding set $\mathcal{A}_i$.

To do this, for each finger, we fix a set of search directions $d^{i,k} \in \mathbb{R}^3$ in the tangent plane at x^i. We then search for the longest step $\theta^{i,k} \geq 0$ that can be taken in this direction while satisfying (9). We denote the point $\theta^{i,k} d^{i,k}$ as $v^{i,k}$ and let $\mathcal{A}_i$ be the convex hull of these points. Since the set on the right side of (9) is convex, $\mathcal{A}_i$ is a conservative approximation.

Conveniently, this search can be expressed as a linear program. Define the matrix $\overline{W} \in \mathbb{R}^{6\times n_w}$ whose columns are the mean basis wrenches. Then, to compute each $v^{i,k}$, we can solve the following LP, denoted VLP^{i,k}:

$$\begin{aligned} \underset{\substack{\theta^{i,k}\in\mathbb{R} \\ \{\alpha_j^{i,k}\}_{j=1}^{n_s}\subset\mathbb{R}^{n_w}}}{\text{maximize}} \quad & \theta^{i,k} && (11a) \\ \text{subject to} \quad & \theta^{i,k} \geq 0 && (11b) \\ & \alpha_j^{i,k} \succeq 0,\ \forall j \in \mathcal{J} && (11c) \\ & \mathbb{1}_{n_w}^\top \alpha_j^{i,k} = 1,\ \forall j \in \mathcal{J} && (11d) \\ & T_j^i\left(\theta^{i,k} d^{i,k}\right) = -\overline{W}\alpha_j^{i,k},\ \forall j \in \mathcal{J}. && (11e) \end{aligned}$$

Constraint (11b) enforces nonnegativity of the scaling along $d^{i,k}$, constraints (11c) and (11d) enforce that the $\alpha_j^{i,k}$ are valid convex weights, and constraint (11e) enforces that the random normal $n^i = \bar{n}^i + \theta^{i,k} d^{i,k}$ must lie in $\mathcal{A}_i$.

Proposition 3. *VLP^{i,k} has a feasible solution if $0 \in \mathcal{C}_{\bar{w}}$.*

Proof. If $0 \in \mathcal{C}_{\bar{w}}$, then $\exists \tilde{\alpha} \in \mathbb{R}^{n_w}$ such that $\tilde{\alpha} \succeq 0, \bar{W}\tilde{\alpha} = 0$, and $\mathbb{1}_{n_w}^\top \tilde{\alpha} = 1$. Thus, the choices $\delta^{i,k} = 0$ and $\alpha_j^{i,k} = \tilde{\alpha},\ \forall j$ satisfy (11b)-(11e), so (11) is feasible. □

Proposition 3 ensures that if a grasp is force closure in the mean case, then we can feasibly solve a batch of VLPs in parallel over all $(i,k) \in \mathcal{I} \times \mathcal{K}$ and then recover a nontrivial value for L_{fc} using the expressions in Proposition 1. If any VLPs in the batch are infeasible, we set the bound to 0 and do not compute any integrals.

IV. Applying PONG to Synthesize Curvature-Regularized Grasps

A common failure mode for grasp synthesis is *edge-seeking*, in which a grasp optimizer computes a solution with the fingertips on sharp edges of the object. We hypothesize two causes. First, in the case of, e.g., the Ferrari-Canny metric, when the torque origin lies near the object's center, the edges yield larger moment arms, theoretically providing more robustness. Second, if an initial guess for a grasp is far from the object, the edges are often the closest points to the fingertips, and edge points are often the first feasible points

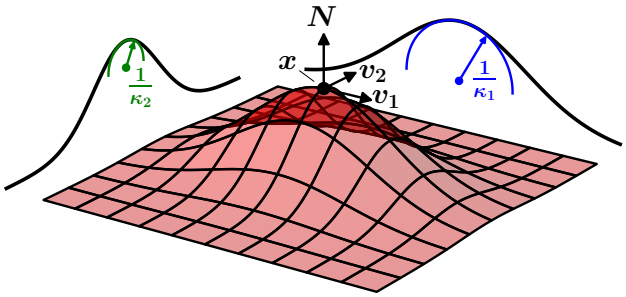


Fig. 5: A visualization of the shape operator. Shown is the surface normal N, the principal directions v_1, v_2, and the principal curvatures κ_1, κ_2. Note the close resemblance to our uncertainty parameterization in Fig. 2a.

encountered during optimization; thus, they easily become local minima for the full optimization.

Edges (and more generally high-curvature areas) tend to be poor grasp locations, because small errors in perception lead to failure. In this section, we use PONG to generate *curvature-regularized* grasps by artificially assigning high uncertainty to high-curvature areas.

A. Curvature-Based Uncertainty Distributions

Suppose we represent the surface $\partial\mathcal{O}$ as the 0-level set of a smooth function $s : \mathbb{R}^3 \to \mathbb{R}$. At a point $x \in \partial\mathcal{O}$, the *shape operator* $\mathfrak{S}_x : T_x\partial\mathcal{O} \to T_x\partial\mathcal{O}$ is defined $\mathfrak{S}_x(v) := -\nabla_v N(x)$, where N is the unit surface normal at x. The two eigenpairs of $\mathfrak{S}_x$, $(\kappa_1, v_1), (\kappa_2, v_2) \in \mathbb{R} \times T_x\partial\mathcal{O}$, are the *principal curvatures and directions* at x respectively [24].

Letting the principal directions v_1, v_2 form our tangent basis at each contact x^i and (some function of) the magnitude of the principal curvatures κ_1, κ_2 form our variances, we can recover a curvature-sensitive uncertainty distribution.

Since $\mathfrak{S}_x(v) = -\left[\nabla^2 s(x)\right](v)$, the eigenpairs of $\mathfrak{S}_x$ are those of the Hessian of s. In the sequel, we use the following uncertainty distribution parameters:

$$\begin{aligned} \bar{n}^i &:= -\nabla s(x^i), \\ \bar{t}^i_m &:= v_m,\ m \in \{1,2\}, \\ \left(\sigma^i_m\right)^2 &:= \log\left(K_{\text{curv}} \cdot |\kappa_m| + \epsilon\right),\ m \in \{1,2\}, \end{aligned} \tag{12}$$

where $K_{\text{curv}} > 0$ is a parameter relating curvature to uncertainty and $\epsilon > 0$ captures prior uncertainty at x. These values are also defined for points $x \notin \partial\mathcal{O}$, i.e., the distribution is even defined "off-surface," which we exploit during optimization to evaluate infeasible iterates.

B. Grasp Synthesis with Nonlinear Optimization

As a test, we synthesized grasps by solving the following bilinear optimization program (similar to FRoGGeR [20]):

$$\begin{aligned} \underset{q \in \mathcal{Q}}{\text{maximize}} \quad & L_{\text{fc}}(q) && \text{(13a)} \\ \text{subject to} \quad & q_{min} \preceq q \preceq q_{max} && \text{(13b)} \\ & \bar{\ell}^*(q) \geq 0.3 && \text{(13c)} \\ & s(FK^i(q)) = 0,\ i = 1, \ldots, n_c && \text{(13d)} \\ & \sigma\left(o_A^{(m)}, o_B^{(m)}; q\right) \geq d_m,\ m = 1, \ldots, n_p. && \text{(13e)} \end{aligned}$$

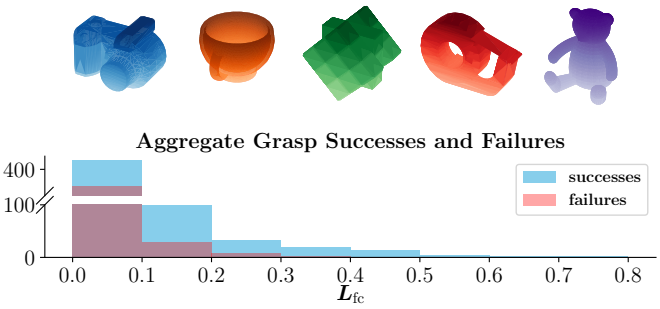


Fig. 6: Simulation results. **Top.** The 5 objects used for simulations (camera, teacup, Rubik's cube, tape dispenser, teddy bear). They were chosen due to their high nonconvexity and many edges. **Bottom.** Success/failure histograms over all trials. Note the fast decay of failures relative to successes.

In (13c), $\bar{\ell}^*(q)$ is the *normalized min-weight metric* [20]. If $\bar{\ell}^*(q) > 0$, then $0 \in \mathcal{C}_{\bar{w}}$, so whenever (13c) is satisfied, we can invoke Proposition 3 to feasibly solve all VLPs.

Constraint (13d) enforces that the contacts x^i lie on the surface, where s is a neural implicit surface trained using volume rendering [25]. Lastly, σ is a signed distance between two convex collision geometries, d_m is the minimum allowable distance between geometries in pair m, and n_p is the total number of collision pairs (see [20] for details).

These pairs are computed by performing a convex decomposition of the object using VHACD [26]. Unlike [20], we *do not* prescribe contact points x^i, so when the fingers are not on the surface, we let x^i be the closest point on the fingertip geometry to the object, computed using `Pinocchio` [27].

C. Simulation Results

Based on the theory of Sec. II, we expect fewer grasp failures as L_{fc} rises. To test this, we synthesized grasps on each of 5 nontrivial objects (see Fig. 6) and visualized the distribution of successes and failures on a "shaky pickup" task, where we added high-frequency sinusoidal perturbations to a straight reference trajectory with an amplitude of $3\texttt{cm}$. We simulated 100 unperturbed and 100 perturbed grasps per object for a total of 1000 grasps.

We use the following simple controller: for an optimal solution q^* of (13), we have corresponding fingertip positions x^i and mean normals $\bar{n}^i$ computed using (12). We define new target positions for each finger $1\texttt{cm}$ into the surface,

$$p^i_{\text{new}} = x^i + 0.01\nabla s(x^i), \tag{14}$$

and let q^*_{new} satisfy $p^i_{\text{new}} = FK^i(q^*_{\text{new}}), \forall i \in \mathcal{I}$, computed using inverse kinematics. Lastly, we track q^*_{new} with P control:

$$\tau_{\text{fb}} = -K_p(q - q^*_{\text{new}}). \tag{15}$$

Shown in Fig. 6 are the aggregate successes and failures over all 1000 grasps. We highlight two observations: (1) as L_{fc} increases, the failure rate decays as predicted; and (2) the bound's conservativeness is nontrivial, as even in the regime where $L_{\text{fc}} \approx 0$, about half of the grasps are successful. That is, if L_{fc} is high, we should be confident in a grasp's quality, but the converse does not hold.

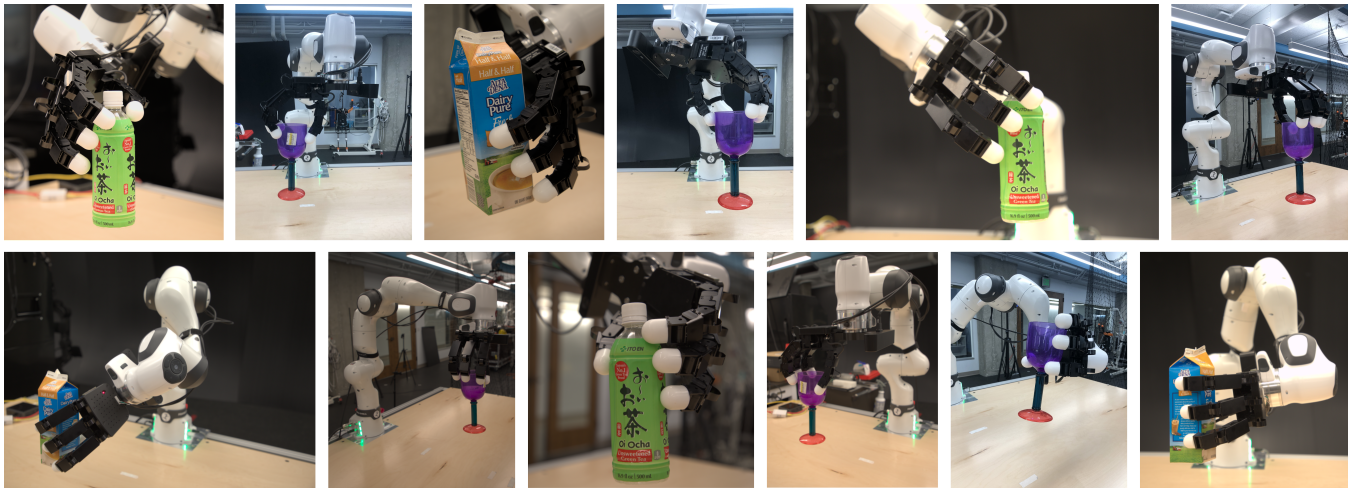


Fig. 7: We used an Allegro hand with an attached Zed camera to image each object and deployed PONG to successfully synthesize diverse precision grasps even in the presence of transparency, reflectivity, and an abundance of edges. Shown are representative successful picks.

We report an aggregate success rate of 59%, a low mark that increases to 75% (266/354) if we exclude very low-confidence grasps ($L_{\text{fc}} < 0.05$). We find that after these exclusions, the remaining grasps are resistant to perturbations, with a success rate of 76.8% (136/177) when unperturbed and 73.4% (130/177) when perturbed, even though aggregately, the perturbations induced a 6.4% drop in success rate. This suggests that many of the induced failures occur in the low-confidence regime, which supports the hypothesis that curvature regularization improves grasp robustness.

Finally, over all trials, the median grasp synthesis time was 5.93s, a result of our optimized implementation of PONG, particularly via parallel LP solves.

D. Hardware Results

We additionally verified that we could use PONG to generate grasps on a real robotic system by synthesizing 10 grasps for each of 3 objects: a semi-transparent tea bottle, an opened (empty) milk carton, and a transparent/reflective colored goblet. To achieve this, we chose to represent each object as a *neural radiance field* (NeRF) [28], from which we extracted a smooth density field defined for any query point $x \in \mathbb{R}^3$. We then represented the surface $\partial\mathcal{O}$ as a level set of this density function, where the chosen level was selected experimentally for each object. The images used to train each NeRF were collected by a wrist-mounted camera which captured images of the object by following a fixed trajectory. We then used this noisy representation to solve problem (13), yielding an optimal grasp configuration q^*.

Finally, we planned a pick trajectory to achieve the optimized grasp. We report pick success rates of 6/10, 7/10, and 7/10 respectively, which we qualify with the following remarks. First, the NeRF representations exhibited significant noise, often with many portions of the object surface missing due to limited views of each object. Second, we deployed the naive "open-loop" grasp controller from Sec. IV-C, which usually fails in the presence of large perception errors. Finally, pick success was very sensitive to the choice of initial condition for the optimizer. For example, all but one failure on the tea bottle were attempted overhead grasps which tried to pinch the poorly-reconstructed cap, while all successes were side grasps. When the initial guess was similar to a good grasp, our experiments support the hypothesis that PONG will refine it into a performant, feasible one.

V. Conclusion

In this paper, we considered the problem of grasp synthesis under surface normal uncertainty. We developed PONG, a novel, analytic lower bound on the probability of force closure (PFC), which provides a principled measure of a grasp's robustness, is fast to compute, and is differentiable. Thus, it is able to serve as an objective function for a gradient-based nonlinear optimizer. We proved that our metric is a lower bound on PFC for Gaussian-distributed surface normals. In particular, we applied PONG to the case of curvature-regularized grasping, where we used the Gaussian curvature of the object surface as an uncertainty metric, and showed that this practical choice of uncertainty distribution yields PFC bounds that are strongly correlated with grasp success/failure. Finally, we provide a hardware study of our method, using this curvature metric to optimize risk-sensitive grasps for objects represented as NeRFs.

This work provides numerous directions for future work. One immediate direction is to consider the effect of uncertain contact locations x^i, which is very common with poor sensors. Another is improving the conservative nature of our bound; we make several linearizing approximations that make the bound fast to compute at the cost of weakening it. It would be also interesting to explore data-driven methods for predicting surface normal distributions from, e.g., visual data. Finally, as discussed in Section IV-D, a key limitation of our hardware experiments was the sensitivity of the grasps to the grasp controller itself, which does not reason about the surface uncertainty; we plan to explore tools from robust optimization, combined with the surface normal uncertainty representations developed here, to perform robust grasp force optimization for risk-sensitive grasp control or in-hand manipulation.

REFERENCES

[1] Máximo A. Roa and Raúl Suárez. Grasp quality measures: review and performance. *Autonomous Robots*, 38:65 – 88, 2014.

[2] Z. Li and S.S. Sastry. Task-oriented optimal grasping by multifingered robot hands. *IEEE Journal on Robotics and Automation*, 4(1):32–44, 1988.

[3] Daniel Kappler, Jeannette Bohg, and Stefan Schaal. Leveraging big data for grasp planning. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4304–4311, 2015.

[4] Umit Rusen Aktas, Chaoyi Zhao, Marek Kopicki, Ales Leonardis, and Jeremy L. Wyatt. Deep Dexterous Grasping of Novel Objects from a Single View. *Int. J. Humanoid Robotics*, 19:2250011:1–2250011:30, 2019.

[5] Lin Shao, Fábio Ferreira, Mikael Jorda, Varun Nambiar, Jianlan Luo, Eugen Solowjow, Juan Aparicio Ojea, Oussama Khatib, and Jeannette Bohg. UniGrasp: Learning a Unified Model to Grasp with N-Fingered Robotic Hands. *ArXiv*, abs/1910.10900, 2019.

[6] Jonathan Weisz and Peter K. Allen. Pose error robust grasping from contact wrench space metrics. In *2012 IEEE International Conference on Robotics and Automation*, pages 557–562, 2012.

[7] Stanimir Dragiev, Marc Toussaint, and Michael Gienger. Gaussian process implicit surfaces for shape estimation and grasping. In *2011 IEEE International Conference on Robotics and Automation*, pages 2845–2850, 2011.

[8] Jeffrey Mahler, Sachin Patil, Ben Kehoe, Jur van den Berg, Matei Ciocarlie, Pieter Abbeel, and Ken Goldberg. GP-GPIS-OPT: Grasp planning with shape uncertainty using Gaussian process implicit surfaces and Sequential Convex Programming. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4919–4926, 2015.

[9] Cristiana de Farias, Naresh Marturi, Rustam Stolkin, and Yasemin Bekiroglu. Simultaneous tactile exploration and grasp refinement for unknown objects. *IEEE Robotics and Automation Letters*, PP, 02 2021.

[10] Muhammad Sami Siddiqui, Claudio Coppola, Gokhan Solak, and Lorenzo Jamone. Discovering stable robot grasps for unknown objects in presence of uncertainty using bayesian models. In *Towards Autonomous Robotic Systems: 22nd Annual Conference, TAROS 2021, Lincoln, UK, September 8–10, 2021, Proceedings*, page 46–55, Berlin, Heidelberg, 2021. Springer-Verlag.

[11] Miao Li, Kaiyu Hang, Danica Kragic, and Aude Billard. Dexterous grasping under shape uncertainty. *Robotics and Autonomous Systems*, 75:352–364, 2016.

[12] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics. *ArXiv*, abs/1703.09312, 2017.

[13] Qingkai Lu, Kautilya Chenna, Balakumar Sundaralingam, and Tucker Hermans. Planning Multi-Fingered Grasps as Probabilistic Inference in a Learned Deep Network. In *International Symposium of Robotics Research*, 2018.

[14] Qingkai Lu, Mark Van der Merwe, Balakumar Sundaralingam, and Tucker Hermans. Multifingered Grasp Planning via Inference in Deep Neural Networks: Outperforming Sampling by Learning Differentiable Models. *IEEE Robotics and Automation Magazine*, 27(2):55–65, 2020.

[15] Hongkai Dai, Anirudha Majumdar, and Russ Tedrake. Synthesis and Optimization of Force Closure Grasps via Sequential Semidefinite Programming. In *International Symposium of Robotics Research*, 2015.

[16] Min Liu, Zherong Pan, Kai Xu, Kanishka Ganguly, and Dinesh Manocha. Deep Differentiable Grasp Planner for High-DOF Grippers. *ArXiv*, abs/2002.01530, 2020.

[17] Tengyu Liu, Zeyu Liu, Ziyuan Jiao, Yixin Zhu, and Song-Chun Zhu. Synthesizing Diverse and Physically Stable Grasps With Arbitrary Hand Structures Using Differentiable Force Closure Estimator. *IEEE Robotics and Automation Letters*, 7:470–477, 2021.

[18] Ruicheng Wang, Jialiang Zhang, Jiayi Chen, Yinzhen Xu, Puhao Li, Tengyu Liu, and He Wang. DexGraspNet: A Large-Scale Robotic Dexterous Grasp Dataset for General Objects Based on Simulation. *ArXiv*, abs/2210.02697, 2022.

[19] Albert Wu, Michelle Guo, and C. Karen Liu. Learning Diverse and Physically Feasible Dexterous Grasps with Generative Model and Bilevel Optimization. *ArXiv*, abs/2207.00195, 2022.

[20] Albert H. Li, Preston Culbertson, Joel W. Burdick, and Aaron D. Ames. FRoGGeR: Fast Robust Grasp Generation via the Min-Weight Metric. *ArXiv*, abs/2302.13687, 2023.

[21] Richard M Murray, Zexiang Li, and S Shankar Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[22] Elon Rimon and Joel Burdick. *The Mechanics of Robot Grasping*. Cambridge University Press, 2019.

[23] Naoki Hayashi, Kohei Segawa, and Shigemasa Takai. 2d voronoi coverage control with gaussian density functions by line integration. *SICE Journal of Control, Measurement, and System Integration*, 10(2):110–116, 2017.

[24] B. O'Neill. *Elementary Differential Geometry, Revised 2nd Edition*. Elsevier Science, 2006.

[25] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 2021.

[26] Khaled Mamou and Faouzi Ghorbel. A simple and efficient approach for 3D mesh approximate convex decomposition. In *2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 3501–3504, 2009.

[27] 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 *International Symposium on System Integration (SII)*, 2019.

[28] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *CoRR*, abs/2003.08934, 2020.

[29] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, 2023.

[30] Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019.

APPENDIX

A. Proof of Proposition 1

The proof of Lemma 1 closely follows the one presented in [23, Proposition 1]. First, we recall Green's Theorem.

Theorem 1 (Green's Theorem)**.** *Let $\mathcal{D}$ be a closed region in the plane with piecewise smooth boundary. Let $P(y_1, y_2)$ and $Q(y_1, y_2)$ be continuously differentiable functions defined on an open set containing $\mathcal{D}$. Then,*

$$\begin{aligned}\oint_{\partial\mathcal{D}} P(y_1,y_2)dy_1 + Q(y_1,y_2)dy_2 \\ = \iint_{\mathcal{D}} \left(\frac{\partial Q}{\partial y_1} - \frac{\partial P}{\partial y_2}\right) dy_1 dy_2.\end{aligned} \tag{16}$$

Second, we prove a useful intermediate result.

Lemma 1. *The following holds:*

$$\begin{aligned}&\int \exp(-(ay^2+2by+c))dy \\ &= \frac{1}{2}\sqrt{\frac{\pi}{a}} \exp\left(\frac{b^2-ac}{a}\right) \operatorname{erf}\left(\sqrt{a}y + \frac{b}{\sqrt{a}}\right) + \textit{const.}\end{aligned} \tag{17}$$

Proof. Recall that

$$\operatorname{erf}(y) = \frac{2}{\sqrt{\pi}} \int_0^y \exp(-t^2)dt. \tag{18}$$

We differentiate the RHS of (17) with respect to y and by the Fundamental Theorem of Calculus, we have

$$\begin{aligned}&\frac{1}{\sqrt{a}} \exp\left(\frac{b^2-ac}{a}\right) \cdot \sqrt{a} \exp\left(-\left(\sqrt{a}y + \frac{b}{\sqrt{a}}\right)^2\right) \\ &= \exp\left(\frac{b^2-ac}{a}\right) \exp\left(-\left(ay^2+2by+\frac{b^2}{a}\right)\right) \\ &= \exp(-(ay^2+2by+c)),\end{aligned} \tag{19}$$

which is the integrand of the LHS, proving the claim. □

We are now ready to prove Proposition 1.

Proof of Proposition 1. We have that

$$\begin{aligned}&(y-\mu)^\top \Sigma^{-1}(y-\mu) \\ &= \frac{1}{\sigma_1^2}y_1^2 - \frac{2}{\sigma_1^2}\mu_1 y_1 + \left[\frac{1}{\sigma_2^2}(y_2-\mu_2)^2 + \frac{\mu_1^2}{\sigma_1^2}\right].\end{aligned} \tag{20}$$

Letting

$$\begin{aligned}a &= \frac{1}{2\sigma_1^2}, \\ b &= \frac{-\mu_1}{2\sigma_1^2}, \\ c &= \frac{1}{2}\left[\frac{1}{\sigma_2^2}(y_2-\mu_2)^2 + \frac{\mu_1^2}{\sigma_1^2}\right],\end{aligned} \tag{21}$$

and applying Lemma 1 to the bivariate Gaussian density function $f(y_1, y_2)$, we have

$$\begin{aligned}&\int f(y_1,y_2)dy_1 \\ &= \frac{1}{2\sigma_2\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{y_2-\mu_2}{\sigma_2}\right)^2\right) \operatorname{erf}\left(\frac{y_1-\mu_1}{\sigma_1\sqrt{2}}\right) \\ &\quad + \text{const.}\end{aligned} \tag{22}$$

Applying Theorem 1, we see that for the choice $P = 0$ and $Q = \int f(y_1, y_2)dy_1$,

$$\begin{aligned}&\iint_{\mathcal{D}} f(y_1,y_2)dy_1dy_2 \\ &= \oint_{\partial\mathcal{D}} \frac{1}{2\sigma_2\sqrt{2\pi}} \exp\left(-\frac{(y_2-\mu_2)^2}{2\sigma_2^2}\right) \operatorname{erf}\left(\frac{y_1-\mu_1}{\sigma_1\sqrt{2}}\right) dy_2.\end{aligned} \tag{23}$$

To evaluate the contour integral, we split the contour up into the line segments formed by connecting the M extreme points of $\mathcal{D}$ in counterclockwise order. A point y on the m^{th} segment can be expressed

$$y = (1-r)\begin{bmatrix}y_1^m \\ y_2^m\end{bmatrix} + r\begin{bmatrix}y_1^{m+1} \\ y_2^{m+1}\end{bmatrix},\ r \in [0,1]. \tag{24}$$

Performing this change of variables and letting $y^{N+1} = y^1$,

$$\begin{aligned}&\iint_{\mathcal{D}} f(y_1,y_2)dy_1dy_2 \\ &= \frac{1}{\sigma_2\sqrt{8\pi}} \sum_{m=1}^{M} D^m \int_0^1 A^m(r)B^m(r)dr,\end{aligned} \tag{25}$$

where

$$\begin{aligned}D^m &:= y_2^{m+1} - y_2^m, \\ A^m(r) &:= \exp\left(-\frac{1}{2\sigma_2^2}\left[(1-r)y_2^m + ry_2^{m+1} - \mu_2\right]^2\right), \\ B^m(r) &:= \operatorname{erf}\left(\frac{(1-r)y_1^m + ry_1^{m+1} - \mu_1}{\sigma_1\sqrt{2}}\right).\end{aligned}$$

Finally, noting that $\mathbb{P}[Z \in \mathcal{D}] = \iint_{\mathcal{D}} f(y_1, y_2)dy_1dy_2$ completes the proof. □

B. Efficiently Solving Batches of VLPs

Program (11) suggests solving a batch of $n_f \cdot n_v$ linear programs in parallel to compute the scaling values $\theta^{i,k} \in \mathbb{R}$. Here, we show that it is equivalent to further parallelize the LP computation in the following way.

Proposition 4 (Efficient VLP Batching)**.** *The following equality holds:*

$$\theta^{i,k} = \min_{j=1,\dots,n_s} \theta_j^{i,k}, \tag{26}$$

where (with a slight abuse of notation) $\theta_j^{i,k}$ is the optimal solution to the following LP for a fixed index triple (i, j, k).

$$\begin{aligned}\underset{\theta_j^{i,k}\in\mathbb{R},\ \alpha_j^{i,k}\in\mathbb{R}^{n_w}}{\text{maximize}} \quad & \theta_j^{i,k} && \text{(27a)} \\ \text{subject to} \quad & \theta^{i,k} \geq 0 && \text{(27b)} \\ & \alpha_j^{i,k} \succeq 0 && \text{(27c)} \\ & \mathbb{1}^\top \alpha_j^{i,k} = 1 && \text{(27d)} \\ & \left(\theta^{i,k} d^{i,k}\right) T_j^i = -\overline{W}\alpha_j^{i,k}. && \text{(27e)}\end{aligned}$$

Proof. Follows by inspecting the dual programs. □

To actually solve many LPs in a batched manner, we use a custom port of the `quantecon` implementation of the simplex method, available at the following link: github.com/alberthli/jax_simplex. Because this implementation is in `JAX`, it can be run on both CPU or GPU without any additional modification. Due to certain parts of our computation stack being CPU-bound, we choose to compute the bound serially entirely on CPU. We observed that an `Intel i9-12900KS` CPU typically exhibited about a 20% increase in speed over an `AMD Ryzen Threadripper PRO 5995WX`, which we attribute to speedups in Intel vs. ARM architectures on BLAS routines.

C. Differentiating the PFC Bound

In order to maximize the bound in (4) in a gradient-based nonlinear optimization program, we must compute the gradient of L_{fc} with respect to the robot configuration q. To accomplish this, we need three major components:

- differentiating through the numerical integration scheme used to evaluate the expressions in Proposition 1 with respect to the polygon vertices $v^{i,k}$;
- differentiating through $\text{VLP}^{i,k}$ in (11) (or the more efficient program (27)) with respect to the wrench matrix $W(q)$;
- differentiating through the parameters W, $\bar{n}^i$, $\{\bar{t}_1^i, \bar{t}_2^i\}$, and $\{\sigma_1^i, \sigma_1^i\}$ with respect to the configuration q.

To differentiate through the numerical integration, we use the open source package `torchquad` designed to differentiate numerical quadrature methods. All sub-expressions in Proposition 1 can be implemented in `JAX`, which yields the desired gradient in a straightforward manner.

To differentiate the optimal value of $\delta^{i,k}$ in (27) with respect to the robot configuration q, we use implicit differentiation of the KKT conditions. The analytical gradient can be computed quickly in a nearly identical way to the one used by FRoGGeR, since here we also solve a linear program [20, Prop. 1]. For the the case of quadratic programs and more general programs, see [31].

We compute the gradients with respect to the uncertainty distribution parameters directly using `JAX`. However, due to the special structure of (27), we compute the gradients of W with respect to the distribution parameters completely analytically, which in practice leads to a large speedup. Because deriving these gradients is extremely tedious and involves an inordinate amount of algebraic manipulation, we defer the details to the open-source implementation, along with test code which verifies the correctness of our analytical derivations against the automatically-computed gradients from `JAX`.

D. Details for the Planar Integration

All search directions $d^{i,k}$ are expressed in the local planar coordinates defined by the principal axes of the shape operator in Sec. IV, which immediately yields a diagonal covariance matrix and planar points $v^{i,k}$. In practice, we do not integrate over the points generating the hull. Instead, we integrate over the polygon formed by connecting the points in lines counterclockwise, which may be smaller than the convex hull (for example, in Fig. 4(b), see the point in the bottom left closest to the origin). We could easily quickly check which points lie on the hull boundary and only integrate over those - we choose not to for simplicity of implementation.

E. Additional Experiment Details

The main difference between our simulated and hardware experiments was the procedure in constructing the object model. In simulation, we found that learning neural SDFs using `sdfstudio` [32] was sufficient, because we could carefully control the training data by placing cameras around the object in a sphere. However, upon trying to recover a representation on real hardware, we were unable to reliably generate a coherent surface model.

Therefore, for the hardware experiments, we instead trained a neural radiance field (NeRF) [28], [29] and represented the object surface as an empirically-chosen level set of the learned density function (in contrast with neural SDFs, where the object surface is always the 0-level set). In practice, we found that the quality of the object representation was sensitive to the choice of density level. It was easy to slightly over or undershoot an appropriate value, which led to grasping failures due to the belief that the object was bigger or smaller than in reality.

The picking trajectory optimization was done entirely in `Drake` [30], including collision avoidance constraints. In order to generate the requisite collision geometries associated with the object, we executed the marching cubes algorithm on the chosen density level set to recover a nonconvex mesh, which we then decomposed into convex bodies using VHACD.

F. Acknowledgments

We thank Lizhi (Gary) Yang for help setting up the hardware experiments. We thank Victor Dorobantu for discussions that led to a valid formulation and proof of Proposition 2. We thank Thomas Lew for discussions regarding Hausdorff distance bounds that inspired the condition of Proposition 2. We thank Georgia Gkioxari for discussions about object representations.

Finally, we thank the maintainers of all the open-source software used extensively in this work, including but not limited to `Drake`, `Pinocchio`, `torchquad`, `JAX`, `sdfstudio`, `trimesh`, `nlopt`, `VHACD`, and `quantecon` [33], [34], [32], [35], [36], [37].

References

[31] Brandon Amos and J. Zico Kolter. OptNet: Differentiable Optimization as a Layer in Neural Networks. In *International Conference on Machine Learning*, 2017.

[32] Zehao Yu, Anpei Chen, Bozidar Antic, Songyou Peng, Apratim Bhattacharyya, Michael Niemeyer, Siyu Tang, Torsten Sattler, and Andreas Geiger. Sdfstudio: A unified framework for surface reconstruction, 2022.

[33] Pablo Gómez, Håvard Hem Toftevaag, and Gabriele Meoni. torchquad: Numerical integration in arbitrary dimensions with pytorch. *Journal of Open Source Software*, 6(64):3439, 2021.

[34] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[35] Dawson-Haggerty et al. trimesh, 2019.

[36] Steven G. Johnson. The NLopt nonlinear-optimization package, 2011.

[37] Dieter Kraft. A software package for sequential quadratic programming. *Forschungsbericht Deutsche Forschungs und Versuchsanstalt fur Luft und Raumfahrt*, 1988.