

# Task-Oriented Dexterous Hand Pose Synthesis Using Differentiable Grasp Wrench Boundary Estimator

Jiayi Chen<sup>1,2</sup>, Yuxing Chen<sup>1,3</sup>, Jialiang Zhang<sup>1,3</sup> and He Wang<sup>1,2,3†</sup>

**Abstract**—This work tackles the problem of task-oriented dexterous hand pose synthesis, which involves generating a static hand pose capable of applying a task-specific set of wrenches to manipulate objects. Unlike previous approaches that focus solely on force-closure grasps, which are unsuitable for non-prehensile manipulation tasks (*e.g.*, turning a knob or pressing a button), we introduce a unified framework covering force-closure grasps, non-force-closure grasps, and a variety of non-prehensile poses. Our key idea is a novel optimization objective quantifying the disparity between the Task Wrench Space (TWS, the desired wrenches predefined as a task prior) and the Grasp Wrench Space (GWS, the achievable wrenches computed from the current hand pose). By minimizing this objective, gradient-based optimization algorithms can synthesize task-oriented hand poses without additional human demonstrations. Our specific contributions include 1) a fast, accurate, and differentiable technique for estimating the GWS boundary; 2) a task-oriented objective function based on the disparity between the estimated GWS boundary and the provided TWS boundary; and 3) an efficient implementation of the synthesis pipeline that leverages CUDA accelerations and supports large-scale parallelizing. Experimental results on 10 diverse tasks demonstrate a 72.6% success rate in simulation. Furthermore, real-world validation for 4 tasks confirms the effectiveness of synthesized poses for manipulation. Notably, despite being primarily tailored for task-oriented hand pose synthesis, our pipeline can generate force-closure grasps 50 times faster than DexGraspNet while maintaining comparable grasp quality. Project page: <https://pku-epic.github.io/TaskDexGrasp/>.

## I. INTRODUCTION

Robotic dexterous hands hold great promise for anthropomorphic manipulation due to their humanoid structure. In recent years, vision-based dexterous grasping methods [1], [2], [3] have seen substantial improvements in generalizability. These successes are empowered by large-scale grasping datasets [4], [5] generated by grasp synthesis algorithms [6], [7], [8]. However, these synthesis algorithms are all limited to generating force-closure grasps, *i.e.*, grasp poses capable of resisting external wrenches from any direction. Nonetheless, synthesizing hand poses for non-force-closure grasping and non-prehensile manipulation is also crucial, especially when a force-closure grasp is unnecessary or unattainable, such as turning a knob or pressing a button.

To address the aforementioned limitation, we propose a unified framework to synthesize dexterous hand poses for various tasks, including force-closure grasping, non-force-closure grasping, and several typical non-prehensile manipulations. Our key insight is to align the Task Wrench Space

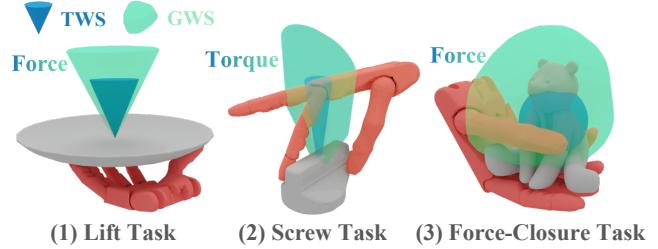


Fig. 1: **Task-oriented dexterous hand poses synthesized for 3 tasks.** Each **TWS** is specified by humans as a task prior: (1) the lift task requires upper forces, (2) the screw task requires counter-clockwise torques, and (3) the force-closure task requires wrenches in all directions. The **GWS** is computed based on the contacts between the hand and the object. Both **TWS** and **GWS** are visualized by their 3D force or torque components.

(TWS) and the Grasp Wrench Space (GWS), where TWS represents the wrench set that the hand *should* apply to the object and GWS represents what the hand *can* apply. Formally, we achieve this by constructing an objective function and minimizing it using gradient-based optimization. This approach can automatically synthesize task-oriented hand poses by simply providing an object mesh and defining a TWS, without requiring additional human demonstrations. Some examples are shown in Fig. 1.

Specifically, our first contribution is a fast, accurate, and differentiable algorithm to estimate the GWS boundary under the max-magnitude ( $L_\infty$ ) bound assumption [9], *i.e.* each contact force is restricted to a maximum value of 1. Our key insight is that, by leveraging the GWS’s specific mathematical properties, we can construct a surjection from 6D unit vectors to its surface, thereby facilitating dense sampling to estimate the GWS boundary (GWB). The computation scales linearly with the contact number  $m$  and sample number  $K$ , and the algorithm is parallelizable with respect to the contacts and samples, enabling scalability in contact-rich scenarios.

Second, we propose a novel task-oriented objective function for gradient-based optimization. This function measures the disparity between the given TWS and the estimated current GWS, thereby encouraging the hand pose to gradually conform to the task specifications during optimization. Formally, we represent the TWS as a 6D hyper-spherical sector parametrized by a 6D unit vector and an angle. In particular, when this angle equals  $180^\circ$ , the TWS degenerates to a 6D hypersphere, indicating the force-closure scenario.

<sup>1</sup>CFCS, School of Computer Science, Peking University.

<sup>2</sup>Beijing Academy of Artificial Intelligence.

<sup>3</sup>Galbot.

Corresponding author: hewang@pku.edu.cn

This formulation of the TWS provides operators with a convenient means to specify task objectives. Nevertheless, our objective function is not limited to this representation.

Finally, we implement the pose synthesis pipeline using cuRobo [10], a motion planning library based on trajectory optimization designed to run on GPUs. Leveraging their CUDA kernels for parallelized forward kinematics, collision checking, and numerical optimization, our pipeline is highly efficient and supports large-scale parallelizing.

Experimental results verify the efficiency and effectiveness of our methods. Our GWS boundary estimator can run in milliseconds on GPU, 200 $\times$  faster than the traditional discretization-based method. Moreover, we design 10 different tasks and achieve an overall success rate of 72.6% in simulation. Real-world validation on 4 tasks further confirms the synthesized poses for manipulation. Notably, despite being primarily tailored for task-oriented hand pose synthesis, our pipeline can synthesize 100,000 force-closure grasps for 5,000 objects within 1.2 GPU hours, 50 $\times$  faster than DexGraspNet [4], with comparable grasp quality.

In summary, our main contributions are (i) a fast, accurate, and differentiable algorithm to estimate the Grasp Wrench Space boundary, (ii) task-oriented objective function for gradient-based hand pose synthesis, and (iii) a heavily optimized dexterous hand pose synthesis pipeline that runs 50 times faster than DexGraspNet.

## II. RELATED WORK

**Task-Oriented Grasp Analysis.** TWS is frequently used to describe the wrenches needed for grasping and manipulating objects. Previous studies [11], [12], [13], [14] approximate TWS as either a 6D ellipsoid or a sphere to simplify grasp analysis, but these formulations are limited to force-closure grasps. Only a few studies [15], [16] have looked into non-force-closure cases, typically focusing on using two or three frictionless fingers to grasp objects with polygonal shapes. Our work defines TWS as a 6D hyperspherical sector, which allows us to handle both force-closure and non-force-closure scenarios. Moreover, our method can accommodate dexterous hands with four or five fingers and complex object meshes.

**Grasp Wrench Space Estimation.** GWS is the basis for most grasp quality metrics [17]. Estimating GWS commonly involves calculating the convex hull over discretized friction cones assuming a sum-magnitude ( $L_1$ ) constraint, where all contact forces sum up to 1 [18]. However, this assumption can be overly conservative and thus unsuitable for multi-finger dexterous hands [9]. Conversely, under the  $L_\infty$  assumption, the conventional discretization-based method exhibits exponential time complexity with respect to the number of contacts [18]. Some studies [19], [20], [14] have utilized optimization techniques to accelerate computation under the  $L_\infty$  assumption, but these approaches are non-differentiable with respect to hand pose. We propose an efficient and differentiable algorithm to approximate the GWS boundary under the  $L_\infty$  assumption.

**Force-Closure Dexterous Grasp Synthesis.** Dexterous grasp synthesis involves both analytical and data-driven methods. Early reviews on this topic can be found in [21], [22]. Analytical methods focus on optimizing certain quality metrics, such as the  $\epsilon$  metric, to discover effective grasps. Previous approaches, like *GraspIt!* [23], rely on sampling-based techniques, which are inefficient for dexterous hands with a high degree of freedom. More recent studies [6], [8], [24] employ gradient-based optimization with differentiable approximations of the  $\epsilon$  metric. However, these approximations often lack strong physical interpretations and are limited to force-closure grasps. Another approach [7], [5] synthesizes dexterous grasps using a differentiable simulator; however, the simulator's gradient may be inaccurate [25].

Data-driven methods [26], [27] rely on learning from data. They often utilize analytical methods for data preparation [4], [5] and post-processing [28], [2] because collecting extensive human data is costly and network predictions may lack accuracy. Recent advancements in dexterous grasping have embraced reinforcement learning [29], [30] and imitation learning [31]. While RL can achieve satisfying results without synthetic datasets, its efficacy is augmented when combined with such data, as exemplified by [1]. This emphasizes the ongoing significance of hand pose synthesis.

**Non-Prehensile Manipulation.** Non-prehensile manipulation encompasses various tasks, including pushing [32], catching [33], pivoting [34], and in-hand manipulation [35]. Given their significant differences from force-closure grasps, prior studies [36], [37] often employ different strategies than conventional grasp synthesis algorithms. In contrast, we present a unified framework for generating both force-closure grasps and various non-prehensile hand poses, including pushing, pulling, lifting, turning, etc.

## III. PRELIMINARIES

Consider an object grasped by a hand with  $m$  contacts. For each contact  $i \in [1, 2, \dots, m]$ , let  $\mathbf{p}_i \in \mathbb{R}^3$  be the contact position,  $\mathbf{n}_i \in \mathbb{R}^3$  be the inward-pointing surface unit normal, and  $\mathbf{d}_i, \mathbf{e}_i \in \mathbb{R}^3$  be two unit tangent vectors such that  $\mathbf{n}_i = \mathbf{d}_i \times \mathbf{e}_i$ , all defined in the object coordinate frame. Although our method also works for the soft contact (SFC) model (please refer to the Appendix), in our main paper, we use the point contact with friction (PCF) model:

$$\mathcal{F}_i^{PCF} = \{ \mathbf{f}_i \in \mathbb{R}^3 \mid 0 \leq f_{i1}, f_{i2}^2 + f_{i3}^2 \leq \mu^2 f_{i1}^2 \} \quad (1)$$

$$\mathbf{G}_i^{PCF} = \begin{bmatrix} \mathbf{n}_i & \mathbf{d}_i & \mathbf{e}_i \\ \mathbf{p}_i \times \mathbf{n}_i & \mathbf{p}_i \times \mathbf{d}_i & \mathbf{p}_i \times \mathbf{e}_i \end{bmatrix} \in \mathbb{R}^{6 \times 3} \quad (2)$$

Here,  $\mathcal{F}_i$  represents all potential forces generated by contact  $i$ , and the matrix  $\mathbf{G}_i$  maps the force set  $\mathcal{F}_i$  to the wrench set  $\mathcal{W}_i = \mathbf{G}_i \mathcal{F}_i$ . We denote  $\mathbf{f}_i = [f_{i1}, f_{i2}, f_{i3}]$  in the  $\langle \mathbf{n}_i, \mathbf{d}_i, \mathbf{e}_i \rangle$  coordinate frame.  $\mu$  is the friction coefficient. The resulting wrench that the robot hand can apply to the object is given by  $\mathbf{w} = \sum_{i=1}^m \mathbf{G}_i \mathbf{f}_i$ , where  $\mathbf{f}_i \in \mathcal{F}_i$ .

All possible  $\mathbf{w}$  constitute the *Grasp Wrench Space* (GWS). In this work, we employ the  $L_\infty$  assumption to calculate the GWS:  $\mathcal{W}_g = \bigoplus_{i=1}^m \mathcal{W}_i$ , representing the Minkowski sum of each  $\mathcal{W}_i$ . We prefer the  $L_\infty$  assumption over the

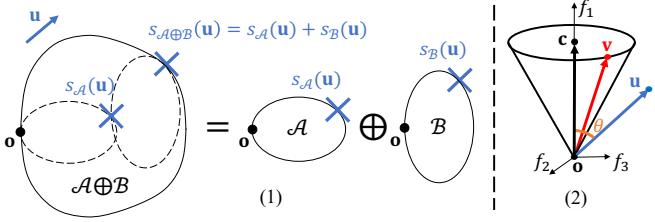


Fig. 2: **Method illustration.** (1) The support mapping  $s_A(\mathbf{u})$  and its Prop. 3. (2)  $s_F(\mathbf{u})$  for PCF contact model in 3D.

$L_1$  assumption due to its greater suitability for multi-finger dexterous hands [9]. *Grasp Wrench Hull (GWH)* and *Grasp Wrench Boundary (GWB)* are defined as the convex hull and the boundary of  $\mathcal{W}_g$ , respectively, denoted as  $\mathcal{W}_g^{ch}$  and  $\partial\mathcal{W}_g$ .

With these notations,  $\epsilon$  metric [18] and its task-oriented extension [14] can be written as

$$\epsilon = \min_{\mathbf{w} \in \partial\mathcal{W}_g} \|\mathbf{w}\|, \quad \epsilon_t = \min_{\mathbf{w} \in \partial\mathcal{W}_g, \mathbf{t} \in \mathcal{W}_t, \mathbf{w} \parallel \mathbf{t}} \frac{\|\mathbf{w}\|}{\|\mathbf{t}\|} \quad (3)$$

Here,  $\mathcal{W}_t$  denotes the TWS,  $\|\cdot\|$  represents the L2 length norm, and  $\parallel$  indicates parallelism.

#### IV. METHOD

##### A. Grasp Wrench Boundary Estimator

Inspired by [19], [38], we propose a novel, fast, accurate, and differentiable algorithm for constructing the GWB through sampling and mapping. The core idea is to map any arbitrary 6D unit direction to a point on the GWB. To achieve this, we begin by defining a support mapping  $s_A$  for a nonempty compact set  $\mathcal{A} \subset \mathbb{R}^n$ :

$$s_A(\mathbf{u}) = \arg \max_{\mathbf{a} \in \mathcal{A}} \mathbf{u}^T \mathbf{a}, \quad \|\mathbf{u}\| = 1 \quad (4)$$

As illustrated in Fig. 2(1),  $s_A(\mathbf{u})$  yields the elements in  $\mathcal{A}$  with the largest projection onto the given unit direction  $\mathbf{u} \in \mathbb{R}^n$ . This mapping also has four useful properties [39], [19]:

*Property 1:*  $\forall \mathbf{u}, s_A(\mathbf{u}) \in \partial\mathcal{A}$ .

*Property 2:* If  $\mathcal{A}$  is convex,  $\forall \mathbf{x} \in \partial\mathcal{A}, \exists \mathbf{u}, \text{s.t. } s_A(\mathbf{u}) = \mathbf{x}$ .

*Property 3:*  $s_{A \oplus B}(\mathbf{u}) = s_A(\mathbf{u}) + s_B(\mathbf{u})$ .

*Property 4:*  $s_{C(A)}(\mathbf{u}) = \mathbf{C} \cdot s_A(\mathbf{C}^T \mathbf{u})$ , where  $\mathbf{C} \in \mathbb{R}^{p \times n}$  and  $p$  is any positive integer. (proved in the appendix)

Here, Prop. 1 indicates that the mapping  $s_A(\mathbf{u})$  always yields an element on  $\partial\mathcal{A}$ , which is the boundary of the set  $\mathcal{A}$ . Prop. 2 shows that if the set  $\mathcal{A}$  is convex, then  $s_A(\mathbf{u})$  is a surjection, indicating that any point on  $\partial\mathcal{A}$  can be mapped. In our scenario, where  $\mathcal{F}_i$ ,  $\mathcal{W}_i$ , and  $\mathcal{W}_g$  are all convex, every point on the GWB can be mapped by  $s_{\mathcal{W}_g}$ . Despite the complexity of  $s_{\mathcal{W}_g}$ , we can simplify it by Prop. 3 and 4:

$$s_{\mathcal{W}_g}(\mathbf{u}) = s_{\bigoplus_{i=1}^m \mathcal{W}_i}(\mathbf{u}) = \sum_{i=1}^m s_{\mathcal{W}_i}(\mathbf{u}) = \sum_{i=1}^m \mathbf{G}_i s_{\mathcal{F}_i}(\mathbf{G}_i^T \mathbf{u}). \quad (5)$$

where  $\bigoplus$  is the Minkowski sum and  $m$  is the contact number.

If we acquire  $s_{\mathcal{F}_i}$ , we can readily deduce  $s_{\mathcal{W}_g}$ . In the context of the PCF contact model, each  $\mathcal{F}_i$  represents a 3D cone. Since our GWS is defined under the  $L_\infty$  assumption,

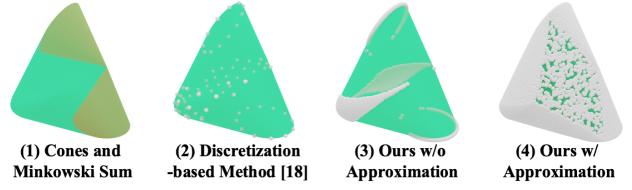


Fig. 3: **GWS estimation visualized in 3D force space.**

(1) In this example, GWS is the Minkowski sum of two cones. (2,3,4) Points (white) are sampled on GWB (green) by different methods. (4) Our method with approximation can get dense samples.

$\mathcal{F}_i$  becomes a finite cone with a height of 1. We can use geometric intuition to directly determine  $s_{\mathcal{F}_i}$ :

$$s_{\mathcal{F}_i}(\mathbf{u}) = \begin{cases} \{\mathbf{f} \mid f_1 = 1, \mathbf{f} \in \mathcal{F}\}, & \text{if } \theta = 0 \\ \mathbf{v}, & \text{if } 0 < \theta < \alpha \\ \{k\mathbf{v} \mid 0 \leq k \leq 1\}, & \text{if } \theta = \alpha \\ \mathbf{o}, & \text{if } \alpha < \theta \leq \pi \end{cases} \quad (6)$$

As shown in Fig. 2(2),  $\mathbf{v}$  represents the intersection between the circle  $\{\mathbf{f} \mid f_1 = 1, f_2^2 + f_3^2 = \mu^2\}$  and the plane spanned by  $\mathbf{c}$  and  $\mathbf{u}$ , where  $\mu$  is the friction coefficient, and  $\mathbf{c} = (1, 0, 0)$ . We also denote  $\theta = \angle(\mathbf{c}, \mathbf{u})$  and  $\alpha = \angle(\mathbf{c}, \mathbf{v}) + \frac{\pi}{2} = \arctan(\mu) + \frac{\pi}{2}$ .

Combining Eq. 5 and 6, we achieve the desired mapping from any arbitrary 6D unit direction to points on the GWB without any approximation. To obtain dense points  $\{\mathbf{w}_k\}$  ( $k = 1, 2, \dots, K$ ) on  $\partial\mathcal{W}_g$ , we only need to uniformly sample  $K$  ( $K$  is a hyperparameter)  $\mathbf{u}_k$  and map them using  $s_{\mathcal{W}_g}$ . This process does not involve optimization and can be fully paralleled for different contacts and samples. Additionally, this approach reduces the exponential time complexity of the classic discretization-based method [18] for the contact number to linear complexity.

**Differentiability.**  $s_{\mathcal{W}_g}(\mathbf{u})$  is differentiable with respect to the contact position  $\mathbf{p}_i$  and normal  $\mathbf{n}_i$  when  $s_{\mathcal{F}_i}(\mathbf{G}_i^T \mathbf{u}) \neq 0$ . This is because  $s_{\mathcal{W}_g}(\mathbf{u})$  in Eq. 5 is differentiable with respect to the matrix  $\mathbf{G}_i$  when  $s_{\mathcal{F}_i}(\mathbf{G}_i^T \mathbf{u}) \neq 0$ . Moreover, the matrix  $\mathbf{G}_i$  in Eq. 2 is differentiable with respect to the contact position  $\mathbf{p}_i$  and normal  $\mathbf{n}_i$ .

**Approximation.** One limitation of the mapping  $s_{\mathcal{F}_i}$  in Eq. 6 is that, when we randomly sample the unit vector  $\mathbf{u}$ , there is an extremely low probability that it will result in  $\theta$  being either 0 or  $\alpha$ . As a result, certain areas on the GWB cannot be mapped, as illustrated in Fig. 3(3). To alleviate this issue, we propose to use an approximation:

$$s_{\mathcal{F}_i}(\mathbf{u}) = \begin{cases} \mathbf{c}, & \text{if } \theta = 0 \\ \mathbf{c} + \theta/\delta \cdot (\mathbf{v} - \mathbf{c}), & \text{if } 0 < \theta < \delta \\ \mathbf{v}, & \text{if } \delta \leq \theta \leq \alpha - \delta \\ (\alpha - \theta)/\delta \cdot \mathbf{v}, & \text{if } \alpha - \delta < \theta < \alpha \\ \mathbf{o}, & \text{if } \alpha \leq \theta \leq \pi \end{cases} \quad (7)$$

Here,  $\delta$  is a hyperparameter that governs the extent of the approximation. The basic idea is to loosen the equality conditions and interpolate between neighboring cases. Notably, we avoid relaxing around the origin  $\mathbf{o}$  to prevent potential

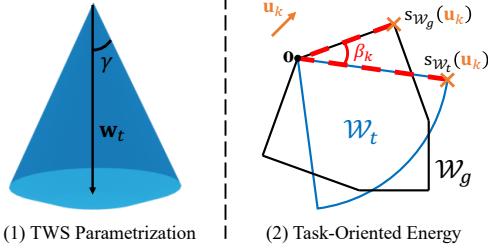


Fig. 4: **Task-oriented energy.** (1) TWS is formulated as a 6D hyper-spherical sector parametrized by a 6D unit vector  $\mathbf{w}_t$  and an angle  $\gamma$ . (2) The task-oriented energy is the sum of the cosine distance between  $s_{\mathcal{W}_t}(\mathbf{u}_k)$  and  $s_{\mathcal{W}_g}(\mathbf{u}_k)$ .

impacts on sampled points near the origin, maintaining metric accuracy. Because the  $\epsilon$  metric can be computed as the smallest magnitude of sampled points on the GWS.

**Contact Position Normalization.** Another challenge is the variability in the L2 norm of the contact position  $\mathbf{p}_i$ . As the L2 norm of the contact normal  $\mathbf{n}_i$  is always 1, if  $\|\mathbf{p}_i\|$  is significantly smaller than  $\|\mathbf{n}_i\|$ , the torque space's magnitude becomes much smaller than that of the force space. In such cases, the GWS is a 6D flat ellipsoid. However, only when the GWS is a 6D sphere can evenly distributed  $\mathbf{u}$  be mapped to evenly distributed points on the GWS. If the GWS deviates too much from a sphere, the sampled points on the GWS will be unevenly distributed, affecting grasp synthesis.

To alleviate this problem and achieve frame invariance, we employ the transformation  $\mathbf{p}'_i = \frac{1}{d}(\mathbf{p}_i - \bar{\mathbf{p}})$ , similar to [19], to normalize the contact position  $\mathbf{p}_i$  before Eq. 2. Here,  $\bar{\mathbf{p}} = \frac{1}{m} \sum_{i=1}^m \mathbf{p}_i$  and  $d = \frac{1}{m} \sum_{i=1}^m \|\mathbf{p}_i - \bar{\mathbf{p}}\|_2$ .

#### B. Differentiable Task-Oriented Energy

As illustrated in Fig. 4, the TWS is formulated as a 6-dimensional hyper-spherical sector:

$$\mathcal{W}_t(\mathbf{w}_t, \gamma) = \{\mathbf{t} \in \mathbb{R}^6 \mid \angle(\mathbf{t}, \mathbf{w}_t) \leq \gamma, \|\mathbf{t}\| = 1\} \quad (8)$$

Here,  $\mathbf{w}_t \in \mathbb{R}^6$  can represent either the average external perturbations to resist or the average active wrenches to apply. Moreover, the angle  $\gamma \in \mathbb{R}$  can denote the tolerance to the potential disturbances, estimation errors, and external force fluctuations during task execution. A larger  $\gamma$  implies a more robust grasp. When  $\gamma = \pi$ , the TWS becomes a hypersphere, representing the force closure task. In this work, we do not delve into the optimal choice of  $\mathbf{w}_t$  and  $\gamma$  for each task [13], and we assume they are given as task priors.

For the energy design, optimizing the metric  $\epsilon_t$  in Eq. 3 directly would be ideal. However,  $\epsilon_t$  is not differentiable when a wrench direction in TWS is not covered by GWS (*i.e.*,  $\epsilon_t = 0$ ). This non-differentiability arises because  $s_{\mathcal{W}_g}(\mathbf{u})$  lacks differentiability when  $s_{\mathcal{W}_g}(\mathbf{u}) = \mathbf{0}$ . Additionally, when initializing a hand pose randomly, it's highly probable to have some wrench directions in TWS not covered by GWS, making  $\epsilon_t$  unsuitable for optimization.

To enable optimization, irrespective of whether  $\epsilon_t = 0$ , we propose a novel energy based on our GWS estimator. For each randomly sampled  $\mathbf{u}_k$ , the same mapping as in

Eq. 4 is used to calculate an element  $s_{\mathcal{W}_t}(\mathbf{u}_k)$  on TWS. Since our TWS is a hyper-spherical sector, solving  $s_{\mathcal{W}_t}(\mathbf{u})$  is straightforward. As shown in Fig. 4, our energy is:

$$E_t = - \sum_{k=1}^K \cos \beta_k \quad (9)$$

where  $\beta_k = \angle(s_{\mathcal{W}_t}(\mathbf{u}_k), s_{\mathcal{W}_g}(\mathbf{u}_k))$ .  $K$  is the sample number.

The cosine distance-based energy term  $E_t$  strongly correlates with the metric  $\epsilon_t$  introduced in Eq. 3. This connection arises from the shared behavior of  $E_t$  and  $\epsilon_t$ . They both reach their minimum values only when the geometric shape of the GWS precisely aligns with that of the TWS. This similarity is the reason why we choose cosine distance over L2 distance. The L2 distance is influenced by the magnitudes of  $s_{\mathcal{W}_g}(\mathbf{u}_k)$ , potentially altering the minimum point. Using cosine distance ensures a more robust alignment based on geometric similarity rather than magnitude considerations.

#### C. Dexterous Grasp Synthesis Pipeline

Our dexterous grasp synthesis pipeline is built upon cuRobo [10]. Our pipeline takes as input an object mesh  $O$ , task parameters  $\{\mathbf{w}_t, \gamma\}$ , and the expected hand contact points  $\{\mathbf{x}_{rest}^i\}_{i=1}^m$  in the rest hand pose ( $m$  is the contact number). Unlike DexGraspNet [4], our hand contact points  $\{\mathbf{x}_{rest}^i\}$  remain unchanged during optimization. The output variable to optimize is the hand pose  $\mathbf{q}$ , including root rotation, translation, and joint angles. For optimization, we employ gradient descent with greedy line search.

During optimization, the hand pose  $\mathbf{q}$  is used to calculate the transformation of each hand link via forward kinematics. These transformations are then applied to  $\{\mathbf{x}_{rest}^i\}_{i=1}^m$  to obtain the posed hand contact points  $\{\mathbf{x}^i\}_{i=1}^m$ . Next, these points are utilized to calculate the contact points  $\{\mathbf{p}^i\}$  and normals  $\{\mathbf{n}^i\}$  by finding the nearest points on the object mesh. To approximate the derivation of the nearest-point calculation, we use finite differences.

The optimization has a total of four energy terms. The first one is the task-oriented energy in Section IV-B, computed using the contact points  $\{\mathbf{p}^i\}$  and normals  $\{\mathbf{n}^i\}$ . The second is the distance energy  $E_d = \sum_{i=1}^m \|\mathbf{x}^i - \mathbf{p}^i\|^2$ , aiming to encourage the hand to make contact with the object. Additionally, we incorporate the penetration and self-penetration energies (denoted as  $E_p$ ) from cuRobo to prevent collisions between the hand and the object.

## V. EXPERIMENTS

This section begins with a sanity check for our proposed GWS estimator. Next, we present quantitative and qualitative results for task-oriented hand pose synthesis across 10 tasks. We also deploy some synthesized poses in the real world. Finally, we compare with DexGraspNet [4] to show our superiority in large-scale force closure grasp synthesis.

#### A. Sanity Check for Grasp Wrench Space Estimation

**Experiment setup.** To validate the efficiency of our GWS estimator, we experiment with 3 parameters: the number of contact points  $m$ , approximation angle  $\delta$ , and sampling

	5 Contacts				7 Contacts			
	Baseline			Ours	Baseline			Ours
	4	6	8		4	6	8	
RLE $\downarrow$	5.30	2.36	1.26	<b>0.43</b>	6.49	2.78	-	<b>0.70</b>
SP $\downarrow$	0.48	0.42	0.38	<b>0.29</b>	0.36	0.31	-	<b>0.26</b>
t $\downarrow$	4e3	2e4	4e4	<b>20</b>	5e4	2e5	2e6	<b>20</b>

TABLE I: **Sanity check for GWS estimator.** ‘-’ indicates that we did not run it, since it is extremely slow (about 0.5 hours per case).

number  $K$ . For each parameter setting, we randomly sample  $m$  contact points on 49 objects from YCB dataset [40], and use them to calculate GWS. We also experiment with different friction coefficients  $\mu = [0.2, 0.3, 0.5, 1.0]$ , resulting in  $49 \times 4 = 196$  test cases per experiment. All metrics below are averaged over these 196 test cases.

We use the classic discretization-based method [18] under  $L_\infty$  assumption as our baseline and experiment with the discretization number  $d = 4, 6, 8$ . Another two parameters  $\delta$  and  $K$  in our method are set as  $15^\circ$  and  $1e6$ , respectively.

**Metrics.** We evaluate the accuracy, density, and speed by three metrics: 1) **average relative length error** (RLE) (unit:  $10^{-2}$ ) of sampled points  $\mathbf{w}$  to the ground truth boundary in the direction of  $\mathbf{w}$ . The error of each point is  $\max_{q \in \partial \mathcal{W}_g^{gt}} \frac{q-1}{q}$ , which can be viewed as a Second-order Cone Program (SOCP) and solved by [41]. 2) **Sparsity** (SP) (unit: rad), the expectation of the min angle between sampled points  $\mathbf{w}$  and uniformly distributed points on the 6D sphere. Since this metric requires GWS to fully cover the sphere, we filter out non-force-closure test cases by randomly sampling many 6-subsets of  $\{\mathbf{w}^k\}$  as simplices and checking whether all orthogonal wrench bases can be expressed as non-negative linear combinations of vertices of a simplex. We resample contact points until we have 196 test cases. 3) average **time** (t) (unit: ms) for each test case.

**Main results.** Table I demonstrates our superior speed, accuracy, and density compared to the discretization-based baseline. The great speed advantage arises because our method can parallelize on GPU without any coding trick, whereas their QuickHull algorithm (Pyhull package) has to iterate on CPU. As the contact number  $m$  increases, our time increases linearly, whereas theirs increases exponentially. Additionally, our method exhibits denser and more accurate results, because our method can map to any point in the GWS with fewer approximations.

**Hyperparameter analysis.** In Tab. II, we investigate the effect of  $\delta$  and  $K$  on the quality of estimated GWS with 5 contact points. A  $\delta$  of  $0^\circ$  results in accurately sampled points on the GWS, but with uneven distribution.  $\delta = 15^\circ$  is a good choice to balance accuracy and uniformity. Higher  $K$  boosts the density but increases computation time.

### B. Task-Oriented Dexterous Hand Pose Synthesis

**Experiment setup.** As there is no standard benchmark for evaluating task-oriented dexterous grasp synthesis, we gathered 10 different objects and created 10 unique tasks. These tasks involve turning, lifting, pushing, pulling, and

	$\delta$ (with $K = 1e5$ )				K (with $\delta = 15^\circ$ )			
	$0^\circ$	$15^\circ$	$30^\circ$	$45^\circ$	$1e3$	$1e4$	$1e5$	$1e6$
RLE $\downarrow$	<b>0.00</b>	0.42	5.45	19.4	0.43	<b>0.42</b>	<b>0.42</b>	0.43
SP $\downarrow$	0.44	<b>0.36</b>	<b>0.36</b>	<b>0.36</b>	0.55	0.45	0.36	<b>0.29</b>
t $\downarrow$	3.2	3.2	3.2	3.2	1.7	1.9	3.2	19.4

TABLE II: **Hyperparameter analysis** on the quality of estimated GWS with 5 contact points.

precision grasping. We optimize 500 iterations and generate 100 grasps per task.

For each task, three conditions must be specified: Task Wrench Space (TWS), hand contact points, and hand pose initialization. Take the task of screwing a knob as an example, we define the TWS parameters as  $\mathbf{w}_t = [0, 0, 0, 0, 0, 1]$  and  $\gamma = 15^\circ$ . The hand contact points are set to be the farthest spheres in the distal link of the index and thumb fingers (cuRobo requires segmenting the robot into spheres). The initial hand position is set above the knob, with the palm facing downward. A random perturbation is then applied to the hand position and orientation. Initially, all hand joint angles are set to 0.

Based on the above example, we only modify some necessary parts for a different task. This usually involves modifying  $\mathbf{w}_t$ , determining which hand fingers to use, and setting the initial hand position and orientation. Specifically, for precision grasps, we set  $\gamma = 180^\circ$ .

**Metrics.** 1) **Simulation success rate** (SS) (unit:%) in Isaac Gym [42]. During evaluation, the hand pose is fixed, and an external wrench is applied to each object at each simulation step. The task succeeds if the object remains unmoved after simulating 100 steps. We conduct a single test for most tasks with the applied external wrench being negative to the  $\mathbf{w}_t$  of the TWS. In the case of precision grasps, we follow the testing procedure for force-closure grasps in DexGraspNet [4], conducting the test 6 times with 6 different force directions. For tasks involving turning, pushing, and pulling, we use a virtual revolute or prismatic joint for each object to restrict its movement. 2) **Time** (t) (unit: s) for synthesizing 100 grasps on an Nvidia RTX 3090 GPU card. 3) **Memory** (M) (unit: GB) cost on GPU.

**Main results.** In Table III, we compare four methods: (1) *Baseline*, which doesn’t incorporate task-oriented energy. (2) *Ablation1*, utilizing L2 distance as the task-oriented energy instead of cosine distance. (3) *Ablation2*, excluding contact position normalization in Sec. IV-A. (4) *Ours*.

We can see that all three variants of our method consistently outperform the baseline, confirming the effectiveness of our task-oriented energy. Our final version performs equally well on different tasks, while other ablations sometimes fail. The L2 energy suffers from low success rates if the magnitude of the GWS significantly differs from the TWS. For example, when using 4 fingers to pull a drawer, the magnitude of GWS should be approximately 4 times larger than that of the TWS. Moreover, without CPN, the GWS estimator and the task-oriented energy will prioritize the force space over the torque space if the contact position

	$E_d, E_p$	$E_t$	CPN	Turn			Lift		Push/Pull		Precision grasp			Average
				Handle	Knob2	Knob3	Handbag	Plate	Button	Drawer	Lid	Key	USBA	
Baseline	✓			13	39	44	0	25	40	10	38	27	2	23.8
Ablation1	✓	L2	✓	<b>84</b>	60	64	10	80	91	48	<b>97</b>	<b>68</b>	<b>62</b>	66.4
Ablation2	✓	cos		79	64	<b>72</b>	50	24	96	<b>99</b>	86	19	2	59.1
Ours	✓	cos	✓	75	<b>73</b>	58	<b>51</b>	<b>97</b>	<b>97</b>	<b>99</b>	74	62	40	<b>72.6</b>

TABLE III: **Simulation success rate (%) for task-oriented pose synthesis.** Three variants of our method consistently outperform the baseline. Our final version performs equally well on different tasks, while other variants fail in some cases.

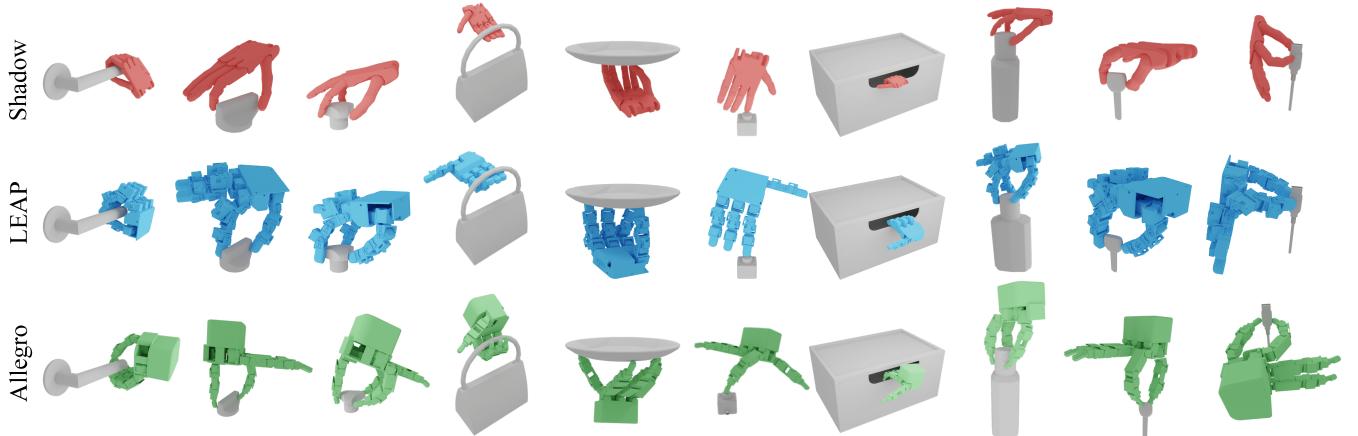


Fig. 5: **Visualization of synthesized task-oriented poses.** We show similar poses for the Shadow hand, but different poses for the Allegro hand.

	$\delta$	$K$	SS (%) $\uparrow$	$t$ (s) $\downarrow$	M (GB) $\downarrow$
Ours	$15^\circ$	$1e2$	72.6	14.9	<b>1.5</b>
Ablation3	$15^\circ$	$1e4$	<b>75.1</b>	83.5	15.3
Ablation4	$0^\circ$	$1e2$	72.1	<b>14.5</b>	<b>1.5</b>
Ablation5	$0^\circ$	$1e4$	74.8	83.4	15.3

TABLE IV: **Hyperparameter analysis** for task-oriented hand pose synthesis.

has a much smaller magnitude than 1. This can significantly impact tasks that require consideration of both force and torque for success. For example, the plate cannot be held stably with a non-zero torque.

**Hyperparameter analysis.** In Table IV, we investigate the impact of two parameters, the approximation angle  $\delta$  and the sampling number  $K$ , in the GWS estimator for grasp synthesis. The default values in Table III are  $\delta = 15^\circ$  and  $K = 1e2$ , striking a balance between speed and success rate. As shown in Table IV, a larger  $K$  improves the average success rate but slows down the synthesis. Leveraging the approximation can slightly improve performance without incurring additional costs.

**Visualization.** Finally, we visualize our synthesized grasps on each task for different robot hands in Fig. 5, such as Shadow hand, LEAP hand [43], and Allegro hand. Additionally, for knob screwing and handle turning, we change the TWS and show 3 diverse grasps per TWS in Fig. 6. Each column shows that different TWS can synthesize different grasps with the same hand pose initialization. Each third grasp is an unnatural example, but the desired force/torque can still be applied.

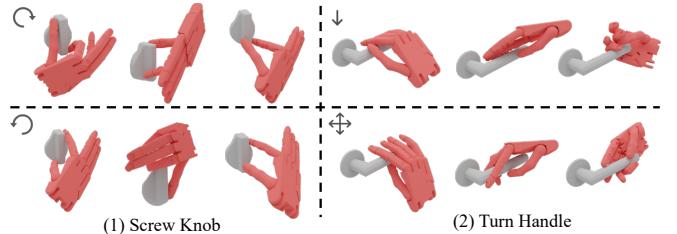


Fig. 6: **Visualization of synthesized poses for different TWS.** Each TWS is denoted by the upper left arrow, while the fourth TWS means force closure.

### C. Real World Experiment for Task-Oriented Poses

Next, we deploy some synthesized task-oriented hand poses in the real world for manipulation. The hardware setup includes a LEAP hand [43] mounted on a UR-5e arm and a Kinect V2 as the depth sensor. The object mesh is obtained by scanning, and its pose is estimated using the Iterative Closest Point (ICP) algorithm. To reach the synthesized hand poses, we use the original cuRobo library for motion planning. The following movements to manipulate the object are manually designed for each task. The design principle is to freeze the hand joints and only move the hand root.

In Fig. 7, we show experiments on lid screwing, handle turning, knob screwing, and drawer pulling. Please refer to the supplementary material for the video. Each task is tested in 10 trials with different grasp poses, resulting in success rates of 5/10, 2/10, 6/10, and 8/10, respectively. The handle-turning task has a low success rate because the restoring force is large. We also observe several failure cases when

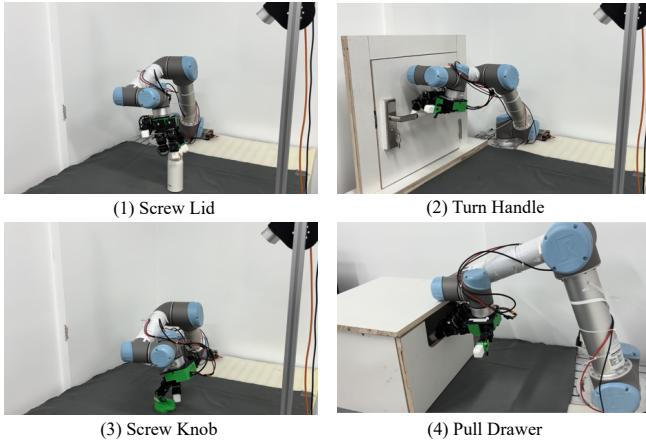


Fig. 7: **Real-world deployments** across 4 tasks.

the finger misses the object by a small margin. Future work can incorporate close-loop tactile feedback to help in.

#### D. Large-Scale Force-Closure Dexterous Grasp Synthesis

**Experiment setup.** To evaluate the synthesis of force-closure dexterous grasps, we use the large-scale object assets from DexGraspNet [4], comprising over 5700 objects. We optimize 500 iterations and generate 20 grasps for each object, resulting in more than 100,000 grasps. The baseline is established using the default settings of DexGraspNet. In our approach, we set  $m = 5$ ,  $\delta = 15^\circ$ , and  $K = 10^2$  to balance the speed and performance. The hand contact points are designated as the farthest spheres in the distal link of each finger. To initialize the hand root pose, we adopt the same approach as DexGraspNet, positioning the hand at a random distance from the object with the palm facing toward it.

**Metrics.** 1) **Simulation success rate** (SS), similar to the one described in Section V-B. 2) **Max penetration depth** (MP) of each grasp. 3)  $\epsilon$  **metric**. Using our GWS estimator, this metric can be calculated as the minimum magnitude of sampled points on GWB. The input to the GWS estimator, contact points, are chosen as the nearest points to the object on each hand link in contact. A link is considered in contact if the distance between a hand link and the object is smaller than 5mm. 4) **times** to synthesize all grasps on a GPU.

**Main results.** As shown in Tab. V, our method can synthesize force-closure grasps significantly faster than DexGraspNet while maintaining comparable quality. This 50x acceleration can be attributed to two primary factors: First, we optimize for only 500 iterations, which is 10 $\times$  fewer than DexGraspNet, yet still yields satisfactory results, thanks to our good objective function. Second, we take advantage of the efficient CUDA functions in the cuRobo library.

We can also find that DexGraspNet synthesizes grasps with a larger  $\epsilon$  metric. This is mainly because their synthesized hands are often closer to the object, leading to more hand links being considered in contact. However, this proximity comes with the drawback of increased penetration compared to our approach.

	SS (%) $\uparrow$	MP (mm) $\downarrow$	$\epsilon \uparrow$	t (hour) $\downarrow$
DexGraspNet	37.0	7.9	<b>0.67</b>	60.0
Ours	<b>42.5</b>	<b>4.8</b>	0.50	<b>1.2</b>

TABLE V: **100k force-closure dexterous grasp synthesis.**

## VI. LIMITATIONS

First, the current formulation can only synthesize a static pose but not a whole trajectory for complex manipulation tasks, *e.g.*, in-hand reorientation. Second, the contact between the object and the environment is not modeled. Finally, employing a geometric primitive hyper-spherical sector to parameterize TWS may not be suitable for all tasks. Future research could explore the use of our method on more complex and long-horizon manipulation tasks.

## VII. CONCLUSIONS

This work proposes a unified framework for efficient task-oriented dexterous hand pose synthesis without human data. Our first contribution is a novel, fast, accurate, and differentiable approach to estimating the GWS. Based on it, we propose a novel task-oriented energy for optimizing dexterous hand poses for various tasks, including non-force-closure grasps, force-closure grasps, and non-prehensile manipulations. Extensive experiments verify the efficiency and effectiveness of our novel GWS estimator, task-oriented energy, and the improved synthesis pipeline.

## REFERENCES

- [1] Y. Xu, W. Wan, J. Zhang, H. Liu, Z. Shan, H. Shen, R. Wang, H. Geng, Y. Weng, J. Chen, *et al.*, “Unidexgrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 4737–4746.
- [2] P. Li, T. Liu, Y. Li, Y. Geng, Y. Zhu, Y. Yang, and S. Huang, “Gendexgrasp: Generalizable dexterous grasping,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 8068–8074.
- [3] J. Lu, H. Kang, H. Li, B. Liu, Y. Yang, Q. Huang, and G. Hua, “Ugg: Unified generative grasping,” *arXiv preprint arXiv:2311.16917*, 2023.
- [4] R. Wang, J. Zhang, J. Chen, Y. Xu, P. Li, T. Liu, and H. Wang, “Dexgraspnet: A large-scale robotic dexterous grasp dataset for general objects based on simulation,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11 359–11 366.
- [5] D. Turpin, T. Zhong, S. Zhang, G. Zhu, J. Liu, R. Singh, E. Heiden, M. Macklin, S. Tsogkas, S. Dickinson, *et al.*, “Fast-grasp’d: Dexterous multi-finger grasp generation through differentiable simulation,” *arXiv preprint arXiv:2306.08132*, 2023.
- [6] T. Liu, Z. Liu, Z. Jiao, Y. Zhu, and S.-C. Zhu, “Synthesizing diverse and physically stable grasps with arbitrary hand structures using differentiable force closure estimator,” *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 470–477, 2021.
- [7] D. Turpin, L. Wang, E. Heiden, Y.-C. Chen, M. Macklin, S. Tsogkas, S. Dickinson, and A. Garg, “Grasp’d: Differentiable contact-rich grasp synthesis for multi-fingered hands,” in *European Conference on Computer Vision*. Springer, 2022, pp. 201–221.
- [8] A. H. Li, P. Culbertson, J. W. Burdick, and A. D. Ames, “Frogger: Fast robust grasp generation via the min-weight metric,” *arXiv preprint arXiv:2302.13687*, 2023.
- [9] R. Krug, Y. Bekiroglu, and M. A. Roa, “Grasp quality evaluation done right: How assumed contact force bounds affect wrench-based quality metrics,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1595–1600.

- [10] B. Sundaralingam, S. K. S. Hari, A. Fishman, C. Garrett, K. V. Wyk, V. Blukis, A. Millane, H. Oleynikova, A. Handa, F. Ramos, N. Ratliff, and D. Fox, “curobo: Parallelized collision-free minimum-jerk robot motion generation,” 2023.
- [11] Z. Li and S. S. Sastry, “Task-oriented optimal grasping by multifingered robot hands,” *IEEE Journal on Robotics and Automation*, vol. 4, no. 1, pp. 32–44, 1988.
- [12] S. El-Khoury, R. De Souza, and A. Billard, “On computing task-oriented grasps,” *Robotics and Autonomous Systems*, vol. 66, pp. 145–158, 2015.
- [13] Y. Lin and Y. Sun, “Grasp planning to maximize task coverage,” *The International Journal of Robotics Research*, vol. 34, no. 9, pp. 1195–1210, 2015.
- [14] C. Borst, M. Fischer, and G. Hirzinger, “Grasp planning: How to choose a suitable task wrench space,” in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 1. IEEE, 2004, pp. 319–325.
- [15] H. Kruger and A. F. van der Stappen, “Partial closure grasps: Metrics and computation,” in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 5024–5030.
- [16] H. Kruger, E. Rimon, and A. F. van der Stappen, “Local force closure,” in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 4176–4182.
- [17] M. A. Roa and R. Suárez, “Grasp quality measures: review and performance,” *Autonomous robots*, vol. 38, pp. 65–88, 2015.
- [18] C. Ferrari, J. F. Canny, *et al.*, “Planning optimal grasps.” in *ICRA*, vol. 3, no. 4, 1992, p. 6.
- [19] Y. Zheng and W.-H. Qian, “Improving grasp quality evaluation,” *Robotics and Autonomous Systems*, vol. 57, no. 6–7, pp. 665–673, 2009.
- [20] Y. Zheng, “An efficient algorithm for a grasp quality measure,” *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 579–585, 2012.
- [21] A. Sahbani, S. El-Khoury, and P. Bidaud, “An overview of 3d object grasp synthesis algorithms,” *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 326–336, 2012.
- [22] J. Bohg, A. Morales, T. Asfour, and D. Kragic, “Data-driven grasp synthesis—a survey,” *IEEE Transactions on robotics*, vol. 30, no. 2, pp. 289–309, 2013.
- [23] A. T. Miller and P. K. Allen, “Graspit! a versatile simulator for robotic grasping,” *IEEE Robotics & Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.
- [24] H. Dai, A. Majumdar, and R. Tedrake, “Synthesis and optimization of force closure grasps via sequential semidefinite programming,” *Robotics Research: Volume 1*, pp. 285–305, 2018.
- [25] Y. D. Zhong, J. Han, and G. O. Brikis, “Differentiable physics simulations with contacts: Do they have correct gradients wrt position, velocity and control?” *arXiv preprint arXiv:2207.05060*, 2022.
- [26] L. Shao, F. Ferreira, M. Jorda, V. Nambiar, J. Luo, E. Solowjow, J. A. Ojea, O. Khatib, and J. Bohg, “Unigrasp: Learning a unified model to grasp with multifingered robotic hands,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2286–2293, 2020.
- [27] W. Wei, D. Li, P. Wang, Y. Li, W. Li, Y. Luo, and J. Zhong, “Dvgg: Deep variational grasp generation for dexterous manipulation,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1659–1666, 2022.
- [28] H. Jiang, S. Liu, J. Wang, and X. Wang, “Hand-object contact consistency reasoning for human grasps generation,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 11 107–11 116.
- [29] P. Mandikal and K. Grauman, “Dexvip: Learning dexterous grasping with human hand pose priors from video,” in *Conference on Robot Learning*. PMLR, 2022, pp. 651–661.
- [30] W. Wan, H. Geng, Y. Liu, Z. Shan, Y. Yang, L. Yi, and H. Wang, “Unidexgrasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning,” *arXiv preprint arXiv:2304.00464*, 2023.
- [31] Y. Qin, Y.-H. Wu, S. Liu, H. Jiang, R. Yang, Y. Fu, and X. Wang, “Dexmv: Imitation learning for dexterous manipulation from human videos,” in *European Conference on Computer Vision*. Springer, 2022, pp. 570–587.
- [32] E. Arruda, M. J. Mathew, M. Kopicki, M. Mistry, M. Azad, and J. L. Wyatt, “Uncertainty averse pushing with model predictive path integral control,” in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 497–502.
- [33] S. Kim, A. Shukla, and A. Billard, “Catching objects in flight,” *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1049–1065, 2014.
- [34] S. Chen, A. Wu, and C. K. Liu, “Synthesizing dexterous nonprehensile pregrasp for ungraspable objects,” in *ACM SIGGRAPH 2023 Conference Proceedings*, 2023, pp. 1–10.
- [35] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal, “Visual dexterity: In-hand reorientation of novel and complex object shapes,” *Science Robotics*, vol. 8, no. 84, p. eadc9244, 2023.
- [36] X. Cheng, S. Patil, Z. Temel, O. Kroemer, and M. T. Mason, “Enhancing dexterity in robotic manipulation via hierarchical contact exploration,” *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 390–397, 2023.
- [37] T. Pang, H. T. Suh, L. Yang, and R. Tedrake, “Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models,” *IEEE Transactions on Robotics*, 2023.
- [38] S. Qiu and M. R. Kermani, “A new approach for grasp quality calculation using continuous boundary formulation of grasp wrench space,” *Mechanism and Machine Theory*, vol. 168, p. 104524, 2022.
- [39] S. R. Lay, *Convex sets and their applications*. Courier Corporation, 2007.
- [40] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, “Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes,” *arXiv preprint arXiv:1711.00199*, 2017.
- [41] A. Domahidi, E. Chu, and S. Boyd, “ECOS: An SOCP solver for embedded systems,” in *European Control Conference (ECC)*, 2013, pp. 3071–3076.
- [42] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, *et al.*, “Isaac gym: High performance gpu-based physics simulation for robot learning,” *arXiv preprint arXiv:2108.10470*, 2021.
- [43] K. Shaw, A. Agarwal, and D. Pathak, “Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning,” *arXiv preprint arXiv:2309.06440*, 2023.

## APPENDIX

### A. Proof for Property 4

$$\begin{aligned} \text{Proof: } s_{\mathbf{C}(\mathcal{A})}(\mathbf{u}) &= \arg \max_{\mathbf{Ca} \in \mathbf{C}(\mathcal{A})} \mathbf{u}^T(\mathbf{Ca}) \\ &= \mathbf{C} \cdot \arg \max_{\mathbf{a} \in \mathcal{A}} (\mathbf{C}^T \mathbf{u})^T \mathbf{a} \\ &= \mathbf{C} \cdot s_{\mathcal{A}}(\mathbf{C}^T \mathbf{u}) \end{aligned} \quad \blacksquare$$

### B. GWS Estimator for Soft Contact Model

The  $\mathcal{F}_i$  and  $\mathbf{G}_i$  for the soft contact model (SFC) is:

$$\mathcal{F}_i^{SFC} = \left\{ \mathbf{f}_i \in \mathbb{R}^4 \mid 0 \leq f_{i1}, \frac{f_{i2}^2 + f_{i3}^2}{\mu_1^2} + \frac{f_{i4}^2}{\mu_2^2} \leq f_{i1}^2 \right\} \quad (10)$$

$$\mathbf{G}_i^{SFC} = \begin{bmatrix} \mathbf{n}_i & \mathbf{d}_i & \mathbf{e}_i & \mathbf{0} \\ \mathbf{p}_i \times \mathbf{n}_i & \mathbf{p}_i \times \mathbf{d}_i & \mathbf{p}_i \times \mathbf{e}_i & \mathbf{n}_i \end{bmatrix} \in \mathbb{R}^{6 \times 4} \quad (11)$$

Eq. 6 can be extended to the soft contact model in 4D. Solving  $\mathbf{v}$  in 4D is the main difficulty. The mathematical intuition behind solving  $\mathbf{v}$  in 3D is the Cauchy-Schwarz inequality. Therefore, in 4D, we can have

$$\begin{aligned} \mathbf{u}^T \mathbf{f} &= \sum_{k=1}^4 u_k f_k \\ &\leq u_1 f_1 + \sqrt{\left( u_2^2 + u_3^2 + \frac{u_4^2 \mu_2^2}{\mu_1^2} \right) \left( f_2^2 + f_3^2 + \frac{f_4^2 \mu_1^2}{\mu_2^2} \right)} \\ &\leq \left( u_1 + \sqrt{u_2^2 \mu_1^2 + u_3^2 \mu_1^2 + u_4^2 \mu_2^2} \right) f_1 \end{aligned}$$

The first inequality is tight when

$$\frac{f_2}{u_2} = \frac{f_3}{u_3} = \frac{f_4 \mu_2^2}{u_4 \mu_1^2}. \quad (12)$$

$\mathbf{v}$  is then the interaction between 2D circle  $\{ \mathbf{f} \mid f_1 = 1, \frac{f_2^2 + f_3^2}{\mu_1^2} + \frac{f_4^2}{\mu_2^2} = 1 \}$  and the space that satisfy Eq. 12.