

Co-Designing Tools and Control Policies for Robust Manipulation

Yifei Dong^{1,*}, Shaohang Han^{1,*}, Xianyi Cheng², Werner Friedl³, Rafael I. Cabral Muchacho¹, Máximo A. Roa³, Jana Tumova¹ and Florian T. Pokorný¹

Abstract—Inherent robustness in manipulation is prevalent in biological systems and critical for robotic manipulation systems due to real-world uncertainties and disturbances. This robustness relies not only on robust control policies but also on the design characteristics of the end-effectors. This paper introduces a bi-level optimization approach to co-designing tools and control policies to achieve robust manipulation. The approach employs reinforcement learning for lower-level control policy learning and multi-task Bayesian optimization for upper-level design optimization. Diverging from prior approaches, we incorporate caging-based robustness metrics into both levels, ensuring manipulation robustness against disturbances and environmental variations. Our method is evaluated in four non-prehensile manipulation environments, demonstrating improvements in task success rate under disturbances and environment changes. A real-world experiment is also conducted to validate the framework’s practical effectiveness¹.

I. INTRODUCTION

In robotic manipulation, the design of specialized tools is essential for enhancing performance. The effectiveness of a tool is inherently linked to how it is controlled: the tool’s design influences the control strategy, while the control policy must adapt to the tool’s specific characteristics. This interdependence between tool design and control strategy forms a co-design problem. Despite significant advancements in co-design methods [1]–[4], algorithmic methods for co-design focusing on robust manipulation have received relatively little attention.

Handling real-world uncertainties and disturbances is crucial for dexterous manipulation tasks. As shown in Figure 1, manipulation challenges often stem from disturbances like object movement or shape variations. To tackle these issues, we focus on two key types of robustness. The first addresses step-wise *disturbances* during execution, such as random perturbations applied to the manipulated objects. The second involves slight variations in the task configuration at the start of each rollout, including changes in the object’s shape, which we refer to as *task variation*, similar to [5].

To co-design tools and policies that consider these disturbances and uncertainties, we turn to robustness

* These authors contributed equally.

¹ The authors are with the division of Robotics, Perception and Learning, KTH Royal Institute of Technology, 10044 Stockholm, Sweden. ² The author is with the Department of Mechanical Engineering and Material Science, Duke University, Durham, NC 27708, USA. ³ The authors are with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), 82234 Wessling, Germany. Funded by the European Commission under the Horizon Europe Framework Programme project SoftEnable, grant number 101070600. Contact: {yifeid, shaohang}@kth.se.

¹For more details, visit <https://sites.google.com/view/robust-codedesign/>.

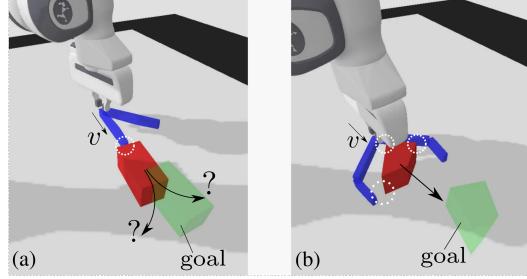


Fig. 1: Strategies for robust manipulation by jointly designing and controlling a tool: Pushing an object towards a goal with only a fingertip contact point is hard. It is easily affected by disturbances or uncertainties (a). Instead, a more robust way is to design a wide-open tool to partially cage the object. By wrapping around it and making potentially more contacts, the tool can “guide” it towards the goal more robustly (b).

evaluation methods for manipulation tasks. Caging is a grasping strategy that allows for non-prehensile manipulation by preventing an object from escaping a tool’s grasp without fully immobilizing it [6]. Compared to prehensile grasps, caging is inherently more robust to geometric variations and uncertainties in position, as the tool’s movement guides the manipulated object without requiring full immobilization in the hand or precise grasping contact points. It follows that caging-based metrics [7], [8] can be used to evaluate the robustness of a given manipulation task.

In this paper, we aim to improve both types of robustness by integrating caging-based metrics, such as partial caging [7] and energy margin scores [8], into the co-design framework. Concretely, we incorporate caging-based metrics as objectives within a bi-level optimization (BLO) framework, a common formulation in co-design problems. To further enhance multi-task efficiency, we employ multi-task Bayesian optimization (MTBO) [9], a global optimization method at the higher level of the BLO framework.

The primary contributions of this work are as follows: (1) We propose a novel approach to tool design for robust manipulation by integrating caging-based robustness metrics, which help manage disturbances and environmental variations. (2) We improve multi-task efficiency by MTBO to optimize tools for general-purpose manipulation across diverse tasks. (3) We validate our approach through extensive experiments in four non-prehensile manipulation environments and demonstrate its effectiveness in a real-world experiment using a Franka Emika Panda robot.

II. RELATED WORK

A. Co-Designing Tools and Control Policies

A common co-design approach involves using a BLO framework [10]–[12], where a global optimization algorithm operates at the higher level, and a control policy, typically trained by reinforcement learning (RL), is at the lower level. Previous studies have explored evolutionary algorithms, such as Genetic Algorithms (GA) [13], at the higher level. However, these methods are often sample inefficient, particularly in multi-task scenarios [14]. In contrast, Bayesian Optimization (BO) and its variant, MTBO, have shown superior efficiency in addressing multi-task problems.

An alternative approach to co-design formulates an integrated Markov decision process (MDP) that encompasses both the design phase and execution phase [14]–[18]. In this framework, the MDP is generally solved through RL, with design and control as sub-policies. However, previous literature focuses solely on task rewards, limiting the flexibility of co-design by restricting the optimization to a single objective. Instead, we maintain the bi-level optimization process that separates design and control, without assuming the differentiability of the dynamics.

B. Robust Manipulation

Robust manipulation has been extensively studied through grasp quality metrics, primarily focused on prehensile tasks. Roa et al. [19] provide a comprehensive review, categorizing classical metrics based on contact configurations and Grasp Wrench Space (GWS) analysis [20]. Key metrics include the GWS volume [21] and the largest resistible disturbance wrench [22]. Task-specific criteria [23] and minimal work for deformable objects [24] further extend these metrics. Machine learning approaches also leverage these analytic metrics for predicting grasp success [25].

However, robustness in non-prehensile manipulation remains under-explored. Caging, introduced by Kuperberg [26], offers a valuable alternative for these scenarios by focusing on preventing object escape through geometric constraints, without relying on force or form closure. Caging concepts include energy-bounded caging [27], which considers external forces like gravity, partial caging [7], etc. Energy margin scores [8] further quantify robustness by evaluating the effort needed for an object to escape. Our work integrates these caging-based metrics into the design and optimization process, providing a more comprehensive approach to robust manipulation across varied scenarios.

III. PROBLEM FORMULATION

In this paper, we aim to co-optimize tool design and control policy for robust manipulation using a bi-level optimization (BLO) framework. This hierarchical structure integrates caging-based manipulation scores to enhance overall robustness and performance.

The BLO optimization framework consists of two nested levels: low-level control policy learning and high-level tool

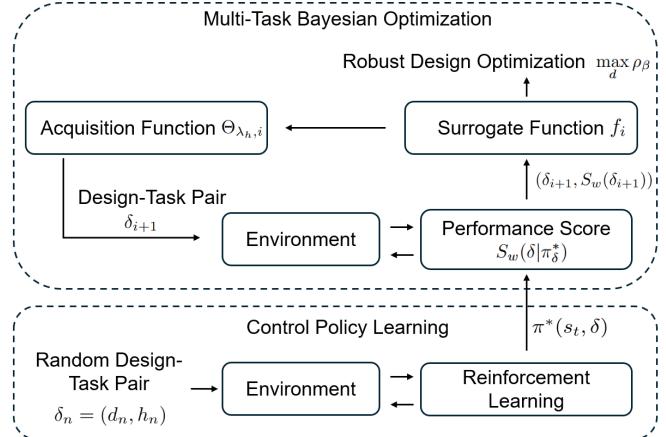


Fig. 2: Structure of the proposed co-design BLO framework.

design optimization:

$$d^* = \arg \max_d \rho_\beta(d, f(\delta, \pi_\delta^*)) \quad (1a)$$

$$\text{s.t. } \pi_\delta^* = \arg \max_\pi \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid \delta \right], \quad (1b)$$

where $\delta = (d, h)$ refers to a design-task pair, composed by a tool design $d \in \mathcal{D}$ and a task configuration $h \in \mathcal{H}$. The design space \mathcal{D} and task configuration space \mathcal{H} differ in each environment, as detailed in Section V-A.

In the lower level, we formulate the control problem as an infinite-horizon MDP with continuous state space, characterized by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, representing the state space \mathcal{S} , action space \mathcal{A} , transition dynamics \mathcal{P} , reward function \mathcal{R} , and discount factor γ . Given a design-task pair δ , the agent executes a policy $\pi_\delta(s_t)$ that selects an action $a_t \in \mathcal{A}$ based on the current state $s_t \in \mathcal{S}$. The environment then computes the next state s_{t+1} and the reward $r_t = R(s_t, a_t, s_{t+1})$. The lower level objective is to learn an optimal control policy π_δ^* that maximizes the expected total discounted reward for a given design-task pair.

The upper level optimizes the design using BO. The objective ρ_β is to find a design that maximizes performance across a range of tasks in \mathcal{H} . To achieve better efficiency, we employ an MTBO approach and use the task-design pair δ as input. As a variant of BO, MTBO utilizes a surrogate function f to approximate the true expensive-to-evaluate performance function. Additionally, MTBO exploits task correlations to further enhance data efficiency.

IV. CO-DESIGNING TOOLS AND CONTROL POLICY

An overview of the proposed method is illustrated in Fig. 2 and Algo. 1. We first train a universal control policy and then employ MTBO at the higher level to find the design that optimizes the performance of robust manipulation.

A. Control Policy Learning

Solving Problem (1) typically requires iteratively optimizing both levels, which involves selecting a design d and training a control policy based on d . This approach can be data inefficient, as training a policy at the lower level

requires interacting with the environment for multiple steps for each trial design. To mitigate this inefficiency, we train a universal control policy $\pi^*(s_t, \delta)$ by solving Problem (1b) and then fix the control policy while optimizing the higher level Problem (1a). By trading specialization for generalization, we need to train only a single policy, significantly improving data efficiency.

We train the policy using actor-critic reinforcement learning, specifically, Proximal Policy Optimization (PPO) [28]. The state space consists of both design and task parameters. Following [14], [15], we employ a dense reward function defined as the weighted sum of three components:

$$R = w_{\text{int}} R_{\text{int}} + w_{\text{suc}} R_{\text{suc}} + w_{\text{rob}} R_{\text{rob}}. \quad (2)$$

The intermediate reward R_{int} , guides the agent towards the goal. The success reward R_{suc} , is awarded as a completion bonus. Lastly, the robustness reward R_{rob} , promotes actions that improve the robustness of the manipulation. For instance, in the catching environment detailed in Section V-A.1, R_{int} penalizes the distance between the current basket position and the predicted object touchdown position, guiding the basket to approach the object. The robustness reward R_{rob} is derived from an energy-bounded caging score, which measures the depth of the object within the basket. In the two pushing environments (Section V-A.2, V-A.3), R_{int} encourages the robot to approach the object, while R_{rob} is based on a partial caging score. The scooping environment (Section V-A.4) employs a dense reward function, incorporating energy margin scores as robustness metrics. The robustness metrics are detailed in Section IV-C.

B. Multi-Task Bayesian Optimization

In the upper level of our framework, we employ MTBO to optimize the design across multiple task configurations. MTBO enables the exploration of both the design space \mathcal{D} and the task configuration space \mathcal{H} by leveraging correlations between tasks.

We first define the *performance score* S_w at a given input configuration δ as

$$S_w(\delta|\pi_\delta^*) = w \mathbb{E}_{s_0}[q_{\text{suc}}(\delta|\pi_\delta^*)] + (1-w) \mathbb{E}_{s_0}[q_{\text{rob}}(\delta|\pi_\delta^*)], \quad (3)$$

where $w \in [0, 1]$ balances success rate q_{suc} and robustness score q_{rob} , evaluated using the learned policy π_δ^* . The expectations are approximately evaluated as the mean cumulative scores for a fixed number of policy rollouts given random initial states $s_0 \in \mathcal{S}$. Since the upper-level optimization is treated as a derivative-free black-box optimization problem, evaluating the input domain $\mathcal{D} \times \mathcal{H}$ by rolling out the lower-level policy is computationally expensive. Therefore, we employ BO, which relies on a probabilistic surrogate model to efficiently approximate the true objective function.

We model the surrogate using a Gaussian Process (GP) to approximate the performance score $S_w(\delta|\pi_\delta^*)$. The *surrogate performance function* is denoted as $f : \delta \mapsto (f_\mu(\delta|\pi_\delta^*), f_k(\delta|\pi_\delta^*))$, which maps a given input δ to its

Algorithm 1: Bi-Level Optimization for Co-Design

```

1 Control Policy Learning (Low-Level Optimization)
Input: Design space  $\mathcal{D}$ , task configuration space  $\mathcal{H}$ ,
initial policy  $\pi_0$ , maximum iterations  $N_{rl}$ .
2 for  $n = 0$  to  $N_{rl}$  do
3   Randomize initial object and robot state  $s_0$ 
4   Sample design  $d_n \in \mathcal{D}$  and task  $h_n \in \mathcal{H}$ 
5   Run policy  $\pi_n(s_t, \delta_n)$  in environment for  $T$  time
steps, where  $\delta_n = (d_n, h_n)$ 
6   Update policy  $\pi_{n+1}$  via PPO
7 end

8 Bayesian Optimization (High-Level Optimization)
Input: Initial design  $d_0 \in \mathcal{D}$ , task  $h_0 \in \mathcal{H}$ ,
 $\delta_0 = (d_0, h_0)$ , maximum iterations  $N_{bo}$ .
9 for  $i = 0$  to  $N_{bo}$  do
10   Select new design-task pair
 $\delta_{i+1} \leftarrow \arg \max_{\delta} \Theta_{\lambda_h, i}(\delta, f_i)$  using acquisition
function  $\Theta_{\lambda_h}$  ▷ EQ. (4), (5)
11   Evaluate the performance score  $S_w(\delta_{i+1})$  using
learned policy  $\pi_\delta$  ▷ EQ. (3)
12   Update GP mean and kernel functions  $f_{\mu, i}, f_{k, i}$ 
with  $(\delta_{i+1}, S_w(\delta_{i+1}))$ 
13 end
14 Compute optimal design  $d_{T_{bo}}^* = \arg \max_d \rho_\beta(d, f_{T_{bo}})$ 
using design score  $\rho_\beta$  ▷ EQ. (6), (7)
15 return Final learned policy  $\pi_\delta^*$ , optimal design  $d^*$ 

```

predicted mean performance score f_μ and the corresponding kernel function f_k . Specifically, we use a multi-task kernel, the intrinsic co-regionalization model (ICM) [29], to model task correlations and improve sample efficiency. For clarity, we omit the dependence on π_δ^* in subsequent expressions.

To guide exploration, we employ a custom Upper Confidence Bound (UCB)-based [30] *acquisition score function* to predict the utility of sampling a particular design-task pair δ based on the current surrogate model f :

$$\Theta_{\lambda_h}(\delta, f) = f_\mu(\delta) + \lambda_h \cdot f_k(\delta), \quad (4)$$

with λ_h a task-specific exploration weight that adjusts the exploration-exploitation trade-off based on the task h . Within MTBO, the acquisition score function is then used as

$$\delta_{i+1} = \arg \max_{\delta} \Theta_{\lambda_h, i}(\delta, f_i) \quad (5)$$

to determine the next best query input δ_{i+1} from the current iteration i . We then evaluate the performance score $S_w(\delta_{i+1})$ of this query and thereby obtain a new data pair $(\delta_{i+1}, S_w(\delta_{i+1}))$. The data pair is used for updating the surrogate mean and kernel functions (f_μ, f_k) .

Given the latest available fit of the surrogate, we determine the optimal design by introducing a *design score function* ρ_β that assigns a score to a given design

$$\rho_\beta(d, f(d, \cdot)) = \beta \cdot \mathbb{E}_{h \sim \mathcal{H}}[f_\mu(d, h)] + (1 - \beta) \cdot \inf\{f_\mu(d, h) \mid h \in \mathcal{H}\}. \quad (6)$$

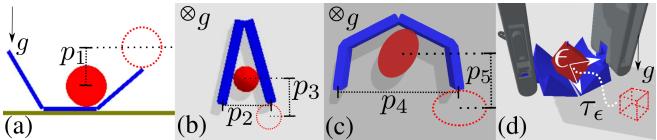


Fig. 3: Illustration of three types of manipulation robustness metrics, detailed in Section IV-C.

The optimal design according to the design score can be obtained by computing

$$d_i^* = \arg \max_d \rho_\beta(d, f_i). \quad (7)$$

The parameter $\beta \in [0, 1]$ controls the trade-off between maximizing average performance ($\beta = 1$) and ensuring robustness against worst-case scenarios ($\beta = 0$). This design score allows us to make risk-aware decisions, balancing performance across all task configurations. The MTBO iteration continues until the maximum number of iterations T_{bo} is reached.

C. Manipulation Robustness Metrics

We integrate robustness metrics into both low-level reward shaping (R_{rob} in Eq. (2)) and high-level design evaluation (q_{rob} in Eq. (3)). Rather than focusing on prehensile grasps, we use caging-based scores for both prehensile and non-prehensile manipulation, including:

1) *Energy-bounded caging score*: Energy-bounded caging score is calculated as the minimum energy an object requires to escape from the end-effector, with a formal definition in [27]. It is exemplified by an object trapped in a basket-like tool under gravity (Fig. 3-a). The score q_{eb} , i.e. the “escape energy”, is given by $q_{eb} = mgp_1$, where m , g represent the mass of the object and the gravitational constant, respectively. Similar concepts are also found in basket grasps [31] and soft fixtures [32].

2) *Partial Caging Score*: An object is considered caged if it cannot escape arbitrarily far [26], while a partial cage allows narrow escape paths in free space. For planar pushing with a V-shaped (Fig. 3-b) or a U-shape (Fig. 3-c) tool, we use an analytical score q_{pc} that computes a weighted average of the tool’s opening width (p_2, p_4) and the object’s depth (p_3, p_5) within the tool, corresponding to the escape path’s length and clearance in [7].

3) *Energy margin score*: Energy margins [8] measure robustness by the effort an object needs to escape a stable configuration, such as the effort an object takes to fall off a scoop (Fig. 3-d). To efficiently compute this, we sample states during manipulation and conduct the energy margin computation by applying random disturbances $\epsilon \sim \mathcal{N}(0, \sigma_{em}^2)$, using a Monte Carlo approach to estimate the average escape time τ_ϵ (similar to [33]). The energy margin score q_{em} is given by $q_{em} = \mathbb{E}_\epsilon[\tau_\epsilon]$. This offers a practical alternative to more complex kinodynamic planning in [8].

V. EVALUATION

We evaluate our method with four manipulation environments depicted in Fig. 4. Our experiments show that incorporating caging-based robustness metrics improves

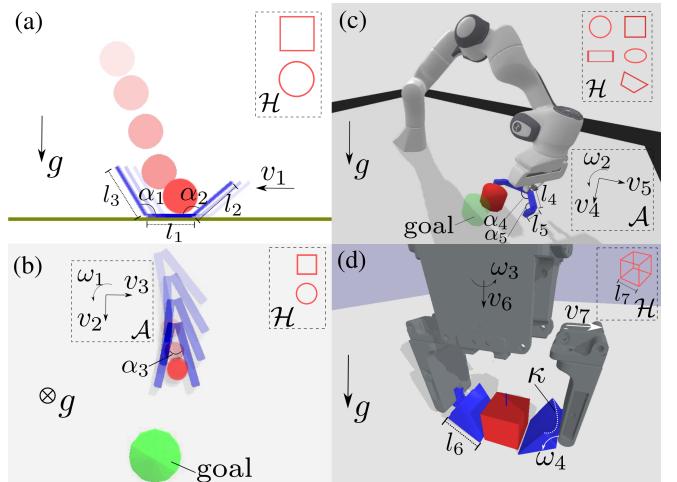


Fig. 4: Four manipulation environments, detailed in Section V-A. The following variables are annotated: action parameters in \mathcal{A} , design parameters in \mathcal{D} and task configurations in \mathcal{H} . Gravity is denoted by g , with the accompanying arrow indicating its direction.

performance under disturbances. The MTBO framework outperforms baseline methods in efficiency and effectiveness. Real-world experiments confirm the successful sim-to-real transfer, demonstrating the practical utility of the optimized designs and policies. We used Pybullet [34] and Box2D [35] for physics simulation.

A. Manipulation Environment

We designed four environments as shown in Fig. 4, spanning 2D and 3D environments, with design spaces \mathcal{D} ranging from 1 to 5 dimensions, discrete or continuous task configuration spaces \mathcal{H} , and action spaces \mathcal{A} ranging from 1 to 6 dimensions.

1) *Catch*: In Fig. 4-a, we design a basket to catch an object falling under gravity with an initial horizontal velocity, with design space $\mathcal{D} = \{l_1, l_2, l_3, \alpha_1, \alpha_2\}$ that parametrizes the length of each segment and the angles between them. The basket moves horizontally with horizontal linear velocity v_1 as the only control action. The task configuration space includes two task objects: $\mathcal{H} = \{\text{circle}, \text{square}\}$. An energy-bounded score evaluates the robustness.

2) *VPush*: This environment involves planar pushing with a symmetric V-shaped hand-held tool (Fig. 4-b). The agent aims to push an object into a circular goal region. The tool’s opening angle is the sole design parameter, $\mathcal{D} = \{\alpha_3\}$. The action space includes the tool’s linear and angular velocities, $\mathcal{A} = \{v_2, v_3, \omega_1\}$. \mathcal{H} contains two task objects: $\mathcal{H} = \{\text{circle}, \text{square}\}$. A partial caging score is used to evaluate robustness.

3) *Panda UPush*: Similar to VPush, this environment designs a symmetric U-shaped tool held by a Franka Panda robot arm (Fig. 4-c) with a design space $\mathcal{D} = \{\alpha_4, \alpha_5, l_4, l_5\}$. Action space includes the tool’s linear v_4, v_5 and angular velocity ω_2 , and the arm is controlled via inverse kinematics. The task objects are all 2.5D, with cross sections from $\mathcal{H} = \{\text{circle}, \text{rectangle}, \text{square}, \text{oval}, \text{irregular quadrilateral}\}$. The robustness is assessed using a partial caging score.

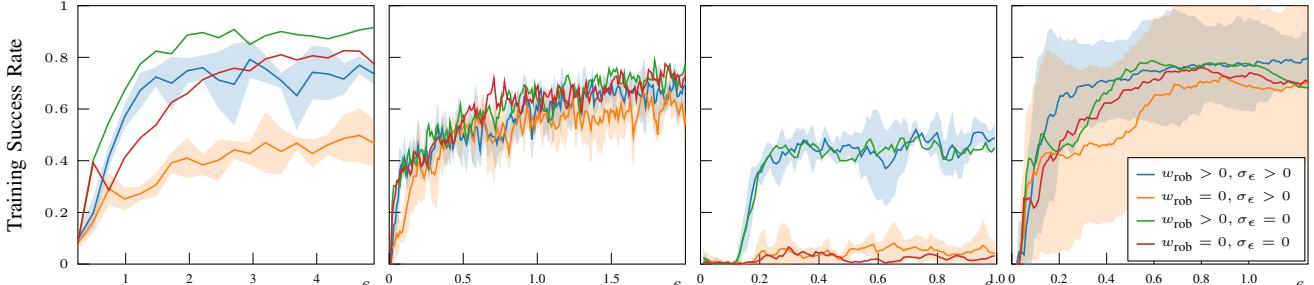


Fig. 5: Success rate over training steps across four environments: Catch, VPush, UPush, and Scoop, respectively. The x-axis represents the number of training steps. To improve visual clarity, the standard deviation shades for curves without training disturbances ($\sigma_\epsilon = 0$) are omitted. The disturbance levels σ_ϵ for each environment are as follows: {0.5, 0.5, 1.8, 0.1}.

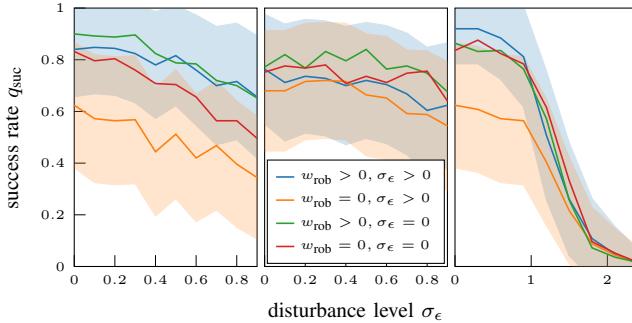


Fig. 6: Success rate across varying disturbance levels σ_ϵ in testing environments (From left to right: Catch, VPush, Scoop.). Each data point represents the average of 250 rollouts (50 rollouts per seed, across 5 random seeds corresponding to the 5 RL models from Fig. 5). For clarity, the standard deviation shading is hidden for two curves as in Fig. 5.

4) *Scoop*: In this environment, we design a scoop gripper's tips to grasp a cube on a table robustly (Fig. 4-d). The agent aims to lift the cube by 0.06 m and maintain it for 1.0 second. The design space \mathcal{D} comprises the fingertip length l_6 and fingertip curvature κ . The action space \mathcal{A} includes the hand vertical linear velocity v_6 , base yaw velocity ω_3 , prismatic finger joint velocity v_7 , and revolute tip joint velocity ω_4 . The task configuration space \mathcal{H} is continuous and 1-dimensional, capturing the cube length l_7 ranging from 0.01 to 0.02 m. An energy margin score is used to evaluate the robustness.

The linear scoop gripper, powered by a QbRobotic QbMove actuator [36], features variable stiffness for adaptive gripping. It employs a double parallelogram mechanism (hidden in Fig. 4-d for clarity) on each side to ensure a parallel closing of the fingers. It uses RC servos to finely adjust the scooping configuration.

B. Numerical Experiment

1) *Robustness Metrics in Policy Learning*: We evaluated the impact of robustness metrics on success rates across four manipulation environments under random disturbances, following [8], [37]. Two variables were controlled: (1) the inclusion of the robustness metric r_{rob} in the RL reward function, where $w_{\text{rob}} > 0$ includes robustness and $w_{\text{rob}} = 0$ excludes it (Eq. (2)); (2) The presence or absence of random disturbance forces $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$ applied to task objects. Fig.

5 illustrates the average success rate curves over five random seeds ($N_{\text{seed}} = 5$) at four combinations of these variables across the manipulation environments. The inclusion of robustness metrics generally gives better performance during policy learning, especially under environmental disturbances. The success rate in the Panda-UPush environment is lower than VPush likely due to the complex design space \mathcal{D} and the demanding goal completion condition. We also observed in Panda-UPush that the robustness metric provides effective guidance, helping the policy avoid poor local optima.

2) *Ablation on Disturbance Levels*: We performed an ablation study to analyze the impact of disturbance intensity σ_ϵ on the RL model performance (Fig. 6). We use control policies trained across 5 random seeds (as shown in Fig. 5) for each environment and combination of decision variables ($\{w_{\text{rob}} > 0, \sigma_\epsilon > 0\}$, $\{w_{\text{rob}} = 0, \sigma_\epsilon > 0\}$, $\{w_{\text{rob}} > 0, \sigma_\epsilon = 0\}$, $\{w_{\text{rob}} = 0, \sigma_\epsilon = 0\}$). The results show a general decline in success rate as disturbance levels increase. Models incorporating robustness metrics ($w_{\text{rob}} > 0$) and disturbances in the training process ($\sigma_\epsilon > 0$) consistently achieve higher success rates compared to those without, even under significant disturbances. This underscores the value of considering robustness in the policy learning process.

3) *Algorithms and Choices in Design Optimization*: We compare the MTBO approach with two baselines: standard BO and GA. For BO, we employ a Lower Confidence Bound acquisition function [38]. For GA, we set a population size of 4 and a mutation rate of 0.1. In both BO and GA, design parameters d are evaluated across all task configurations $h \in \mathcal{H}$, with success rates averaged over multiple rollouts. In contrast, MTBO selects a specific design-task pair $\delta = (d, h)$ and optimizes by leveraging task correlations. In Table I, we evaluate two scenarios: incorporating robustness metrics ($w < 1$) in the performance score S_w (Eq. 3), and excluding them ($w = 1$). Q_{s5} and Q_{e5} represent the confidence threshold of the test success rate, evaluated using the optimal design d_i^* from the first five and last five iterations, respectively: $Q_{s5,\mu} = \sum_{j=1}^{N_{\text{seed}}} \sum_{i<5} \mathbb{E}_{h \sim \mathcal{H}}[q_{\text{suc}}((d_i^*, h) | \pi_{\delta,j}^*)]$, and analogously for $Q_{e5,\mu}$ except $i > T_{\text{bo}} - 5$. Policies $\pi_{\delta,j}^*$ used in the evaluations come from the five random-seeded models trained during the RL phase, as shown in Fig. 5. Note that we either add robustness metrics in both levels ($w_{\text{rob}} > 0$, $w < 1$) or remove them all ($w_{\text{rob}} = 0$, $w = 1$). Rollouts were

TABLE I: Success rate improvement in design optimization of different methods.

Method	w	β	Catch		VPush		UPush		Scoop		\bar{N}_{roll}
			Q_{s5}	Q_{e5}	Q_{s5}	Q_{e5}	Q_{s5}	Q_{e5}	Q_{s5}	Q_{e5}	
MTBO	< 1	0	0.79 ± 0.16	0.89 ± 0.09	$0.81 \pm \mathbf{0.10}$	0.84 ± 0.06	0.42 ± 0.12	$0.46 \pm \mathbf{0.11}$	0.85 ± 0.18	0.88 ± 0.16	650
	< 1	1	0.55 ± 0.32	$0.91 \pm \mathbf{0.06}$	$\mathbf{0.82} \pm 0.11$	$\mathbf{0.85} \pm \mathbf{0.03}$	0.38 ± 0.15	0.53 ± 0.20	0.88 ± 0.14	0.93 ± 0.11	
	$= 1$	1	0.27 ± 0.26	0.67 ± 0.17	0.70 ± 0.15	0.80 ± 0.07	0.10 ± 0.15	0.23 ± 0.17	0.66 ± 0.36	0.71 ± 0.36	
BO	< 1	1	0.73 ± 0.21	$\mathbf{0.92} \pm 0.08$	0.73 ± 0.12	0.83 ± 0.09	0.43 ± 0.12	0.44 ± 0.11	0.88 ± 0.12	0.90 ± 0.07	690
	$= 1$	1	0.26 ± 0.20	0.75 ± 0.13	0.65 ± 0.15	0.83 ± 0.14	0.14 ± 0.16	0.22 ± 0.20	0.76 ± 0.38	0.77 ± 0.39	
GA	< 1	1	$\mathbf{0.88} \pm \mathbf{0.10}$	0.91 ± 0.07	0.78 ± 0.15	0.79 ± 0.12	$\mathbf{0.52} \pm \mathbf{0.08}$	0.53 ± 0.15	$\mathbf{0.89} \pm \mathbf{0.09}$	$0.91 \pm \mathbf{0.10}$	1140
	$= 1$	1	0.61 ± 0.14	0.69 ± 0.11	0.77 ± 0.14	0.82 ± 0.10	0.16 ± 0.19	0.26 ± 0.21	0.75 ± 0.38	0.73 ± 0.38	

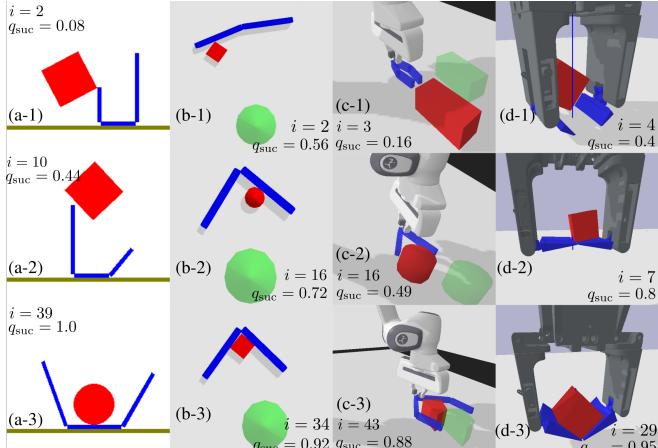


Fig. 7: Design optimization process of the four environments. The iteration number (i), the optimal design at i (d_i^*), and test success rate of the design (q_{suc} , short for $\mathbb{E}_{h \in \mathcal{H}}[q_{\text{suc}}((d_i^*, h)|\pi_{\delta,j}^*)]$) are annotated in the subfigures. Here, the optimizer is MTBO with $w < 1$ and $\beta = 1$.

performed under disturbance conditions ($\sigma_\epsilon > 0$). \bar{N}_{roll} refers to the minimum average number of control policy rollouts to achieve convergence in the design optimization.

The results show that incorporating robustness metrics ($w < 1$) generally leads to higher success rates, emphasizing the benefits of integrating robustness into training and design optimization. GA demonstrated promising results at the first few iterations (high $Q_{s5,\mu}$) at the cost of more design policy rollouts per iteration than MTBO. However, MTBO can achieve comparable or better performance (Q_{e5}) with less number of minimum rollouts \bar{N}_{roll} .

Additionally, we implemented an ablation study on the choice of β in Eq. 6. From the first two rows of Table I, it is demonstrated that maximizing average performance ($\beta = 1$) over the task configuration space \mathcal{H} leads to designs with slightly better performance (Q_{e5}) than maximizing worst-case scenario ($\beta = 0$).

Fig. 7 illustrates the optimal designs identified throughout the optimization process. The results show a clear tendency for the optimization to favor designs with higher robustness scores, which, in turn, significantly improve manipulation success rates under disturbances.

C. Real-world Experiment

We conducted physical evaluation experiments using a Franka Emika Panda robot arm [39] to test the designs d_i^* and policies $\pi^*(s_t, \delta)$ trained in simulation. Three optimized

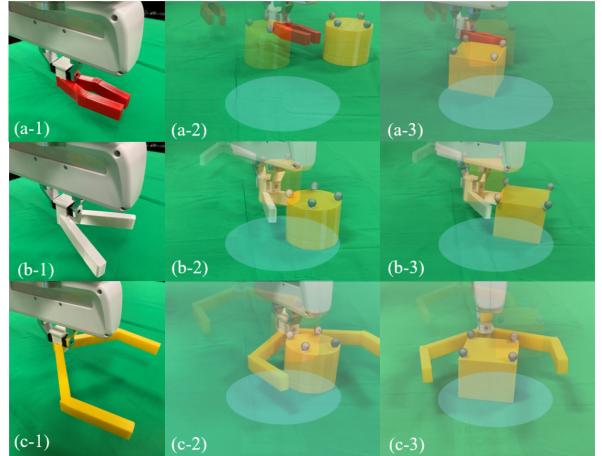


Fig. 8: Real-world experiments. The goal regions are shown in blue.

tools from the upper-level design optimization process (Fig. 7-c) were attached to the end-effector of the arm (Fig. 8). We selected two task objects $\{\text{circle}, \text{square}\}$ from \mathcal{H} . A motion capture system is set up to track the pose of objects. The results in Fig. 8 demonstrate successful sim-to-real transfer of the learned policy. MTBO effectively adapts the designs from a narrow, chopstick-like tool to wide-open tools (Fig. 7, c-1 to c-3). The narrow tool (Fig. 8, a-1) struggles to push the circular object to the goal due to slippage at contact (Fig. 8, a-2), consistent with the failure case shown in Fig. 1-a. In contrast, the wide-open grippers (Fig. 8, b-1, c-1) better handle task variations, successfully pushing objects to the goal. However, sim-to-real challenges remain, including the RL policy’s generalizability to unseen states, positioning errors in the motion capture system, and inaccuracies in the arm’s Cartesian impedance control.

VI. CONCLUSION

We presented steps towards designing tools and control policies for robust object manipulation. Our evaluation in simulation and the real world demonstrated the utility of the robustness metrics in designing and controlling tools in the presence of disturbances and uncertainties. In future work, we aim to showcase the capabilities of this framework in co-designing for robust deformable object manipulation with the scoop gripper.

ACKNOWLEDGMENT

Thanks to Mengyuan Zhao for proofreading and Haofei Lu and Alberta Longhini for assistance with the experiments.

REFERENCES

- [1] J. Xu, T. Chen, L. Zlokapa, M. Foshey, W. Matusik, S. Sueda, and P. Agrawal, “An end-to-end differentiable framework for contact-aware robot design,” in *Robotics: Science & Systems*, 2021.
- [2] T. Chen, Z. He, and M. Ciocarlie, “Hardware as policy: Mechanical and computational co-optimization using deep reinforcement learning,” in *Conference on Robot Learning*. PMLR, 2021, pp. 1158–1173.
- [3] T.-H. J. Wang, J. Zheng, P. Ma, Y. Du, B. Kim, A. Spielberg, J. Tenenbaum, C. Gan, and D. Rus, “Diffusebot: Breeding soft robots with physics-augmented generative diffusion models,” *Advances in Neural Information Processing Systems*, vol. 36, 2024.
- [4] Z. He and M. Ciocarlie, “Morph: Design co-optimization with reinforcement learning via a differentiable hardware model proxy,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 7764–7771.
- [5] M. Li, R. Antonova, D. Sadigh, and J. Bohg, “Learning tool morphology for contact-rich manipulation tasks with differentiable simulation,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1859–1865.
- [6] A. Rodriguez, M. T. Mason, and S. Ferry, “From caging to grasping,” *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 886–900, 2012.
- [7] T. Makapunyo, T. Phoka, P. Pipattanasomporn, N. Niparnan, and A. Sudsang, “Measurement framework of partial cage quality,” in *2012 IEEE international conference on robotics and biomimetics (ROBIO)*. IEEE, 2012, pp. 1812–1816.
- [8] Y. Dong, X. Cheng, and F. T. Pokorny, “Characterizing manipulation robustness through energy margin and caging analysis,” *IEEE Robot. Automat. Lett.*, 2024.
- [9] K. Swersky, J. Snoek, and R. P. Adams, “Multi-task bayesian optimization,” *Advances in neural information processing systems*, vol. 26, 2013.
- [10] J. Bhatia, H. Jackson, Y. Tian, J. Xu, and W. Matusik, “Evolution gym: A large-scale benchmark for evolving soft robots,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 2201–2214, 2021.
- [11] K. S. Luck, H. B. Amor, and R. Calandra, “Data-efficient co-adaptation of morphology and behaviour with deep reinforcement learning,” in *Conference on Robot Learning*. PMLR, 2020, pp. 854–869.
- [12] R. Deimel, P. Irmisch, V. Wall, and O. Brock, “Automated co-design of soft hand morphology and control strategy for grasping,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1213–1218.
- [13] O. Kramer and O. Kramer, *Genetic algorithms*. Springer, 2017.
- [14] Z. Liu, S. Tian, M. Guo, K. Liu, and J. Wu, “Learning to design and use tools for robotic manipulation,” in *7th Annual Conference on Robot Learning*, 2023.
- [15] M. Guo, Z. Liu, S. Tian, Z. Xie, J. Wu, and C. K. Liu, “Learning to design 3d printable adaptations on everyday objects for robot manipulation,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 824–830.
- [16] Y. Yuan, Y. Song, Z. Luo, W. Sun, and K. M. Kitani, “Transform2act: Learning a transform-and-control policy for efficient agent design,” in *International Conference on Learning Representations*, 2022.
- [17] L. Jackson, C. Walters, S. Eckersley, P. Senior, and S. Hadfield, “Orchid: optimisation of robotic control and hardware in design using reinforcement learning,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 4911–4917.
- [18] C. Rajani, K. Arndt, D. Blanco-Mulero, K. S. Luck, and V. Kyrki, “Co-imitation: learning design and behaviour by imitation,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 5, 2023, pp. 6200–6208.
- [19] M. A. Roa and R. Suárez, “Grasp quality measures: review and performance,” *Autonomous robots*, vol. 38, pp. 65–88, 2015.
- [20] N. S. Pollard, “Synthesizing grasps from generalized prototypes,” in *Proc. Int. Conf. Robot. Automat.*, vol. 3. IEEE, 1996, pp. 2124–2130.
- [21] A. T. Miller and P. K. Allen, “Examples of 3d grasp quality computations,” in *Proc. Int. Conf. Robot. Automat.*. IEEE, 1999.
- [22] C. Ferrari, J. F. Canny, et al., “Planning optimal grasps.” in *Proc. Int. Conf. Robot. Automat.*, vol. 3, no. 4, 1992, p. 6.
- [23] Y. Lin and Y. Sun, “Task-based grasp quality measures for grasp synthesis,” in *Proc. Int. Conf. Intell. Robot. Syst.*, 2015.
- [24] J. Xu, M. Danielczuk, J. Ichnowski, J. Mahler, E. Steinbach, and K. Goldberg, “Minimal work: A grasp quality metric for deformable hollow objects,” in *Proc. Int. Conf. Robot. Automat.*, 2020.
- [25] A. Saxena, L. L. Wong, and A. Y. Ng, “Learning grasp strategies with partial shape information.” in *AAAI*, vol. 3, no. 2, 2008, pp. 1491–1494.
- [26] W. Kuperberg, “Problems on polytopes and convex sets,” in *DIMACS Workshop on polytopes*, 1990, pp. 584–589.
- [27] J. Mahler, F. T. Pokorny, S. Niway, and K. Goldberg, “Synthesis of energy-bounded planar caging grasps using persistent homology,” *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 3, pp. 908–918, 2018.
- [28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
- [29] E. V. Bonilla, K. Chai, and C. Williams, “Multi-task gaussian process prediction,” *Advances in neural information processing systems*, vol. 20, 2007.
- [30] A. Garivier and E. Moulines, “On upper-confidence bound policies for switching bandit problems,” in *International conference on algorithmic learning theory*. Springer, 2011, pp. 174–188.
- [31] A. Shirizly and E. D. Rimon, “Selection of secure gravity based caging grasps of planar objects: Robustness and experimental validation,” *IEEE Transactions on Robotics*, 2024.
- [32] Y. Dong and F. T. Pokorny, “Quasi-static soft fixture analysis of rigid and deformable objects,” in *Proc. Int. Conf. Robot. Automat.*. IEEE, 2024, pp. 6513–6520.
- [33] Y. Kim, Z. Pan, and K. Hauser, “Mo-bbo: Multi-objective bilevel bayesian optimization for robot and behavior co-design,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 9877–9883.
- [34] E. Coumans and Y. Bai, “Pybullet, a python module for physics simulation for games, robotics and machine learning,” 2016.
- [35] E. Catto, “Box2d,” <https://box2d.org/>.
- [36] qbrobotics, “qbmove advanced,” 2024, accessed: 2024-04-20. [Online]. Available: <https://qbrobotics.com/product/qbmove-advanced/>
- [37] Y. Shirai, D. K. Jha, and A. U. Raghunathan, “Robust pivoting manipulation using contact implicit bilevel optimization,” *IEEE Transactions on Robotics*, 2024.
- [38] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, “Gaussian process optimization in the bandit setting: No regret and experimental design,” *arXiv preprint arXiv:0912.3995*, 2009.
- [39] S. Haddadin, S. Parusel, L. Johannsmeier, S. Golz, S. Gabl, F. Walch, M. Sabaghian, C. Jähne, L. Hausperger, and S. Haddadin, “The franka emika robot: A reference platform for robotics research and education,” *IEEE Robot. Automat. Mag.*, vol. 29, no. 2, pp. 46–64, 2022.