Policy-guided Sampling-based Predictive Control for Contact-Rich Manipulation

Author Names Omitted for Anonymous Review. Paper-ID 347

***Abstract*—This paper proposes Policy-guided CEM, a novel control framework that combines pre-trained reinforcement learning policies with Sampling-based Model Predictive Control (MPC). The RL policy provides a nominal control distribution learned from extensive simulation experience, while the MPC optimizes residual controls online to minimize a cost function while satisfying constraints. This new approach is applied to non-prehensile manipulation tasks such as pushing, which present significant challenges for robotic control due to discontinuous contact dynamics and underactuation. We demonstrate the performance of the Policy-guided CEM using high-fidelity simulations of the Franka Panda robot pushing various objects to a target pose under parameter uncertainty, and subject to operational and safety constraints.**

***Index Terms*—Non-prehensile manipulation, model predictive control, reinforcement learning, contact-rich manipulation**

I. Introduction

As the complexity of robotic tasks increases, the scope of possible interactions with their environment grows exponentially, posing a theoretical challenge for modeling and control design. Among various robotic manipulation paradigms, non-prehensile manipulation (controlling objects without grasping them) represents a particularly challenging domain due to its reliance on intermittent contacts and complex friction dynamics. These include behaviors such as pushing, sliding, rolling, and juggling, where robots must coordinate their motions with object dynamics through contact forces [1]. Designing model-based controllers for such tasks is difficult because the dynamics are strongly nonlinear and discontinuous whenever contacts bind or break.

Historically, Model Predictive Control (MPC) has served as a cornerstone for addressing optimal control problems in robotic manipulation. By optimizing control inputs over a finite time horizon using a physics-based model, MPC provides a principled framework to explicitly handle critical constraints, such as joint torque limits, obstacle avoidance, and friction cones at the contact interface. However, real-time implementation often requires a delicate trade-off between model fidelity and computational overhead. While simplified models can facilitate basic pick-and-place routines, they often struggle with tasks involving heavy object manipulation or complex, multi-contact coupling between the end-effector and the environment. Accurate control in these scenarios requires expanding the model to consider full-body momentum and articulated joint dynamics [2]. This increased complexity creates a computational bottleneck, necessitating significant algorithmic optimization to maintain the high update rates required for closed-loop stability.

Conversely, Reinforcement Learning (RL) has demonstrated remarkable capabilities in discovering complex, dexterous motor skills through exploratory trial and error. RL policies can learn to exploit intricate full-body dynamics and contact physics that are often difficult to capture in analytical models. Despite these strengths, RL approaches suffer from significant drawbacks: they are notoriously sample-inefficient, lack formal safety or stability guarantees, and operate as black-box models that are difficult to interpret or tune for safety-critical manipulation tasks.

Sampling-based Predictive Control (SPC) has emerged as a prominent alternative to gradient-based MPC for solving nonlinear optimal control problems. By substituting the sophisticated nonlinear optimizers of classic MPC with a parallelizable sampling routine, SPC attains competitive performance across diverse manipulation systems [3]. Representative algorithms include Model Predictive Path Integral Control (MPPI) [4], Predictive Sampling (PS) [5], and the Cross-Entropy Method (CEM) [6]. The conceptual simplicity of SPC, paired with recent advances in high-performance GPU-accelerated simulation [7, 8, 9], has enabled successful deployment on demanding problems such as in-hand dexterous manipulation [3]. On many tasks, SPC matches the performance of state-of-the-art RL methods while requiring no offline policy training [3]. SPC is particularly attractive for contact-rich manipulation, where the discontinuous dynamics produced by frequent making and breaking of contact often hinder general-purpose nonconvex optimizers [10, 11, 12, 13]. Even elementary SPC variants can surpass specialized contact-implicit techniques because they avoid expensive gradient calculations through contact events, as stochastic sampling naturally smooths the underlying nonsmooth dynamics [14, 15].

Despite the power of GPU-accelerated SPC, the curse of dimensionality remains a significant bottleneck for the whole-body control of high-DOF robotic platforms. A dexterous robot may possess 20+ actuators, and sampling a high-dimensional space randomly is inherently inefficient. This motivates the integration of RL and MPC, leveraging their complementary strengths. MPC provides physics-grounded constraint enforcement and local trajectory optimization, while RL offers global guidance, long-horizon value estimation, and the ability to learn complex contact sequences. One common approach to bridge both approaches is to use learning to approximate the infinite-horizon cost-to-go for the MPC [16]. Alternatively, residual approaches employ a policy that learns to provide targeted corrections to the MPC outputs [17]. Conversely, RL-guided sampling strategies utilize a pre-trained policy to bias the sampling distribution toward high-reward

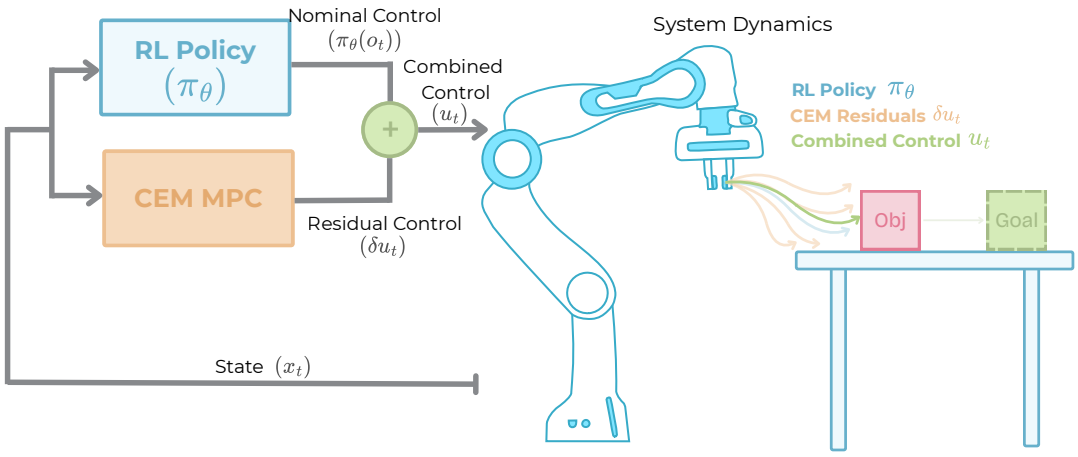


Fig. 1: Policy-guided CEM control architecture.

regions of the state space. Residual-MPPI [18] represents a cutting-edge formulation that integrates Residual Q-Learning with SPC planning, defining the control distribution to minimize divergence from a prior policy while maximizing the task reward.

In this work, we introduce POLICY-GUIDED CEM, a hybrid control architecture that unifies the global task knowledge of Reinforcement Learning with the online planning capability of Sampling-based MPC. Unlike prior residual formulations, our framework leverages the Cross-Entropy Method as a zero-order solver, shown to perform well when faced with discontinuous dynamics inherent to contact-rich tasks [3]. By treating a pre-trained RL policy as an informative prior, the optimizer identifies a risk-aware residual correction that enforces constraints and is robust to model parameter uncertainty.

Our core contributions are as follows:

- **Policy-guided CEM:** A hybrid control framework leveraging the complementary strengths of RL and SPC.
- **Zero-Shot Generalization:** Systematic demonstration of superior adaptability to out-of-distribution object geometries and significant parametric uncertainty (e.g., mass and friction) without the need for retraining.
- **Constraint-Aware Planning:** A robust constraint-handling mechanism that preserves constraint adherence while maintaining high task performance.

II. PRELIMINARIES

This section establishes the theoretical foundations and mathematical notation required to develop the POLICY-GUIDED CEM framework. We begin by formalizing the optimal control problem. Subsequently, we provide an overview of Sampling-based Predictive Control and Reinforcement Learning, the two paradigms whose complementary strengths are unified in our proposed architecture.

A. Problem Statement

In this paper, we consider the infinite-horizon constrained optimal problem

$$J^*_\infty(\mathbf{x}_0) = \min_{\mathbf{u}_{0:\infty}} \quad \sum_{t=0}^{\infty} \ell(\mathbf{x}_t, \mathbf{u}_t) \\ \text{s.t.} \quad \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t), \quad c(\mathbf{x}_t, \mathbf{u}_t) \leq 0, \tag{1}$$

where $\ell(\cdot)$ is the stage cost, $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ denotes the dynamics and $J^*_\infty(\mathbf{x})$ is the Optimal Value Function, representing the minimum cost-to-go from state $\mathbf{x} \in \mathbb{R}^n$. The function $c : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ defines the inequality constraints. Solving (1) directly is often intractable for high-dimensional, contact-rich systems due to the nonlinearity of f and the presence of discontinuous contact dynamics.

B. Sampling-based Predictive Control

Model Predictive Control approximates the infinite-horizon problem (1) by solving a finite-horizon optimization over a horizon T at each time step

$$J^*(\mathbf{x}_0) = \min_{\mathbf{u}_{0:T-1}} \quad \phi(\mathbf{x}_T) + \sum_{t=0}^{T-1} \ell(\mathbf{x}_t, \mathbf{u}_t) \\ \text{s.t.} \quad \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t), \quad c(\mathbf{x}_t, \mathbf{u}_t) \leq 0, \tag{2}$$

where the tail of the cost of the infinite horizon optimal control problem is approximated by the terminal cost $\phi(\cdot)$. For simplicity, we denote the T-length control sequence as $\mathbf{U} = [\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{T-1}]$ and rewrite (2) in compact form

$$\min_U J(\mathbf{U}; \mathbf{x}_0), \tag{3}$$

where both the costs and dynamics constraints are wrapped into the (highly non-convex) objective.

Traditional MPC solves (2) via gradient-based optimization, which tends to perform poorly in the stiff, non-linear contact regimes typical of manipulation [19]. Sampling-based Predictive Control has emerged as a promising derivative-free

alternative [3, 4, 20]. At each control step, SPC draws N candidate control sequences $\mathbf{U}^{(i)} \sim \mathcal{N}(\bar{\mathbf{U}}, \Sigma)$ and rolls out each from the current state $\mathbf{x}_0$ to compute costs

$$J^{(i)} = J(\mathbf{U}; \mathbf{x}_0). \tag{4}$$

The mean sequence is then updated via a weighted average

$$\bar{\mathbf{U}} \leftarrow \bar{\mathbf{U}} + \frac{\sum_{i=1}^{N} g(J^{(i)})(\mathbf{U}^{(i)} - \bar{\mathbf{U}})}{\sum_{i=1}^{N} g(J^{(i)})}, \tag{5}$$

where the weighting function $g(\cdot)$ distinguishes various SPC frameworks. For instance, *Model Predictive Path Integral* (MPPI) control utilizes an exponential weighting

$$g(J) = \exp(-J/\lambda), \tag{6}$$

with a temperature $\lambda > 0$, while *Predictive Sampling* [5] simplifies this to a greedy selection of the single best sample via

$$g(J) = \lim_{\lambda \to 0} \exp(-J/\lambda), \tag{7}$$

Alternatively, the *Cross Entropy Method* (CEM) [6] employs an indicator function

$$g(J) = \mathbf{1}[J \leq \kappa], \tag{8}$$

where κ is defined implicitly by fraction ζ of the top elite samples ranked by lowest cost. CEM also iteratively adapts both the proposal mean and the covariance Σ based on this elite set to narrow the search space toward low-cost regions. Due to its algorithmic simplicity and recent literature highlighting its superior performance in complex robotic manipulation tasks [3, 21], we adopt CEM as the primary SPC approach in this work. The complete CEM procedure is summarized in Algorithm 1.

Algorithm 1 Cross Entropy Method (CEM)

Require: Initial state $\mathbf{x}_0$, iterations K, samples N, horizon T, elite fraction ζ.

for iteration $k = 1$ **to** K **do**
  Sample N control candidates: $\mathbf{U}^{(1\ldots N)} \sim \mathcal{N}(\bar{\mathbf{U}}, \mathbf{\Sigma})$
  for each sample $i \in \{1, \ldots, N\}$ **in parallel do**
    $\mathbf{x}_0^{(i)} \leftarrow \mathbf{x}_0$
    for $t = 0$ **to** $T - 1$ **do**
      $\mathbf{x}_{t+1}^{(i)} \leftarrow f(\mathbf{x}_t^{(i)}, \mathbf{u}_t^{(i)})$
    end for
    $J^{(i)} \leftarrow \phi(\mathbf{x}_T^{(i)}) + \sum_{t=0}^{T-1} \ell(\mathbf{x}_t^{(i)}, \mathbf{u}_t^{(i)})$
  end for
  $\mathcal{E} \leftarrow$ Indices of $\lfloor \zeta N \rfloor$ samples ranked by lowest $J^{(i)}$
  Update: $\bar{\mathbf{U}} \leftarrow \text{mean}(\{\mathbf{U}^{(i)}\}_{i \in \mathcal{E}})$
  Update: $\mathbf{\Sigma} \leftarrow \text{cov}(\{\mathbf{U}^{(i)}\}_{i \in \mathcal{E}})$
end for
Return $\mathbf{u}_0^* = \bar{\mathbf{u}}_0$

C. Reinforcement Learning

Implementing MPC entails solving Trajectory Optimization (TO) problems in real-time, a task often complicated by strict hardware and timing constraints. Although SPC methods have progressed significantly, they generally fall short of the performance levels achieved by Reinforcement Learning (RL). The primary advantage of RL lies in its runtime efficiency, since the computational burden of finding a policy that generalizes across initial conditions is handled during offline training, the live system need only evaluate a pre-trained policy.

RL aims to solve the unconstrained variant of (1) offline by learning an amortized policy $\pi_\theta(\mathbf{x})$ parametrized by a neural network with parameters θ. Algorithms such as Proximal Policy Optimization (PPO) [22] optimize θ to minimize the expected infinite-horizon cost

$$\begin{aligned} J_\theta &= \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t \ell(\mathbf{x}_t, \mathbf{u}_t)\right] \\ \text{s.t.} \quad \mathbf{u}_t &= \pi_\theta(\mathbf{x}_t), \quad \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t), \end{aligned} \tag{9}$$

with a discount factor γ.

III. Methodology: Policy-guided CEM

We now introduce Policy-guided CEM, a framework that combines a pre-trained policy with online planning via the CEM zero-order optimizer. In our approach, π_θ serves as a global, amortized approximate solution to the optimal control problem (1), capturing a high-level understanding of the task manifold that we aim to refine online to minimize the task cost. We further extend this method with a constraint handling mechanism and an optional robustness scheme via online domain randomization.

A. Cost Formulation

In our framework, the decision variable for the trajectory optimization is not the raw control input, but a residual control sequence $\boldsymbol{\delta}\mathbf{U} = \{\boldsymbol{\delta}\mathbf{u}_0, \ldots, \boldsymbol{\delta}\mathbf{u}_{T-1}\}$. We parameterize the control distribution $\mathcal{Q}$ as a state-dependent shift of a pre-trained RL policy $\pi_\theta(\mathbf{x})$, parametrized as Gaussian $\pi_\theta(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}_\pi(\mathbf{x}), \mathbf{\Sigma}_\pi(\mathbf{x}))$, which serves as our prior $\mathcal{P}$. Specifically, the control applied at each step is

$$\mathbf{u}_t = \boldsymbol{\mu}_\pi(\mathbf{x}_t) + \boldsymbol{\delta}\mathbf{u}_t, \tag{10}$$

where $\boldsymbol{\mu}_\pi(\mathbf{x}_t)$ is the mean control action provided by the RL policy. We then seek an optimal trajectory distribution $\mathcal{Q}(\mathbf{u})$ that minimizes the variational free energy

$$\mathcal{J}(\mathcal{Q}) = \mathbb{E}_{\mathbf{U} \sim \mathcal{Q}}\left[J(\mathbf{U}; \mathbf{x}_0)\right] + \eta D_{\text{KL}}(\mathcal{Q}|\mathcal{P}), \tag{11}$$

where D_{KL} denotes the Kullback-Leibler (KL) divergence from the prior policy with η a weighting coefficient. The KL term acts as a regularizer that penalizes deviations from the learned behavior.

Assuming the policy is Gaussian and the distribution $\mathcal{Q}$ is a shifted version of this Gaussian, the KL-divergence term in (11) reduces to a quadratic penalty on the residual. This yields our proposed stage cost formulation

$$\tilde{\ell}(\mathbf{x}_t, \mathbf{u}_t) = \ell(\mathbf{x}_t, \mathbf{u}_t) + \eta \boldsymbol{\delta}\mathbf{u}_t^\top \mathbf{\Sigma}_\pi^{-1}(\mathbf{x}_t) \boldsymbol{\delta}\mathbf{u}_t. \tag{12}$$

This formulation provides an intuitive mechanism for policy-guided search: in state-space regions where the policy is confident (low variance $\mathbf{\Sigma}_\pi$), the Mahalanobis penalty is

high, encouraging the optimizer to stick closely to the prior. Conversely, where the policy is uncertain (high variance), the penalty is low, allowing the local optimizer more freedom to explore corrective residuals to satisfy constraints or minimize costs.

B. Constraint Handling

Standard CEM, detailed in Algorithm 1, does not natively handle constraints $c(\mathbf{x}, \mathbf{u}) \leq 0$. Inspired by [23], we introduce a scoring function S to encourage feasibility over performance.

For each candidate sample i, we compute the accumulated task cost $J^{(i)}$ and the accumulated constraint violation $C^{(i)}$

$$C^{(i)} = \sum_{t=0}^{T-1} \max(0, c(\mathbf{x}_t^{(i)}, \mathbf{u}_t^{(i)})). \tag{13}$$

The feasibility-aware score is defined as

$$S^{(i)} = \begin{cases} J^{(i)}, & \text{if } C^{(i)} = 0, \\ C^{(i)} + J_{\max} + \rho, & \text{if } C^{(i)} > 0, \end{cases} \tag{14}$$

where $J_{\max} = \max\{J^{(i)} \mid C^{(i)} = 0\}$ is the maximum cost among the feasible candidates in the current batch, and $\rho > 0$ is a small positive margin to ensure a strict separation between feasible and infeasible regions. This formulation ensures that any candidate trajectory that satisfies constraints is strictly ranked higher than any infeasible candidate.

C. Algorithm Overview

The complete POLICY-GUIDED CEM algorithm is detailed in Algorithm 2. At each control step, the framework performs local trajectory optimization centered around the RL prior to iteratively refine the residual distribution to minimize task cost and satisfy constraints.

Algorithm 2 Policy-guided CEM

Require: Prior $\pi_\theta(\mathbf{u}|\mathbf{x})$, initial state $\mathbf{x}_0$, horizon T, iterations K, samples N, elite fraction ζ.

for iteration $k = 1$ **to** K **do**
  Sample N residual candidates: $\boldsymbol{\delta}\mathbf{U}^{(1\ldots N)} \sim \mathcal{N}(\bar{\boldsymbol{\delta}}\mathbf{U}, \boldsymbol{\Sigma})$
  for each sample $i \in \{1, \ldots, N\}$ **in parallel do**
    $\mathbf{x}_0^{(i)} \leftarrow \mathbf{x}_0$
    for $t = 0$ **to** $T - 1$ **do**
      $\mathbf{u}_t^{(i)} \leftarrow \boldsymbol{\mu}_\pi(\mathbf{x}_t^{(i)}) + \boldsymbol{\delta}\mathbf{u}_t^{(i)}$
      $\mathbf{x}_{t+1}^{(i)} \leftarrow f(\mathbf{x}_t^{(i)}, \mathbf{u}_t^{(i)})$
    end for
    Compute $J^{(i)}$ and $C^{(i)}$
  end for
  $S^{(i)} \leftarrow \text{Score}(J^{(i)}, C^{(i)})$ via Eq. (14)
  $\mathcal{E} \leftarrow$ Indices of $\lfloor \zeta N \rfloor$ samples ranked by lowest $S^{(i)}$
  Update: $\bar{\boldsymbol{\delta}}\mathbf{U} \leftarrow \text{mean}(\{\boldsymbol{\delta}\mathbf{U}^{(i)}\}_{i \in \mathcal{E}})$
  Update: $\boldsymbol{\Sigma} \leftarrow \text{cov}(\{\boldsymbol{\delta}\mathbf{U}^{(i)}\}_{i \in \mathcal{E}})$
end for
Return $\mathbf{u}_0^* = \boldsymbol{\mu}_\pi(\mathbf{x}_0) + \bar{\boldsymbol{\delta}}\mathbf{u}_0$

D. Extension: Online Domain Randomization

To improve robustness against model mismatch and physical uncertainty, SPC can leverage the massive parallelism of modern GPU simulators to perform online domain randomization. Following the approach outlined in [24], we modify the SPC rollout step to simulate each candidate control sequence $\mathbf{u}^{(i)}$ simultaneously across D parallel physics domains, each with randomized model parameters $\theta_d \sim p(\theta)$ (e.g., varying friction coefficients, masses, or damping).

To perform the CEM update, costs and constraint violations across physics domains must be aggregated into a single scalar value per candidate sequence. We denote this risk-strategy aggregation operator as $\mathcal{A} : \mathbb{R}^D \to \mathbb{R}$. Several strategies for $\mathcal{A}$ exist that offer different trade-offs between average-case performance and risk aversion:

1) Expected Cost: The standard approach in Reinforcement Learning domain randomization is to optimize for the average performance across all sampled environments

$$\mathcal{A}_{\text{avg}}(\mathbf{Y}) = \mathbb{E}_{d \sim p(\theta)}\left[Y^{(d)}\right] \approx \frac{1}{D}\sum_{d=1}^{D} Y^{(d)}. \tag{15}$$

2) Worst-Case Cost: A conservative minimax approach optimizes against the worst-case realization of the dynamics

$$\mathcal{A}_{\text{worst}}(\mathbf{Y}) = \max_{d \in \{1,\ldots,D\}}\left[Y^{(d)}\right]. \tag{16}$$

3) Conditional Value-at-Risk (CVaR): CVaR provides a tunable balance between the expected and worst-case costs by considering the expectation of the $(1 - \beta)$ tail of the distribution

$$\mathcal{A}_{\text{CVaR}}(\mathbf{Y}) = \inf_{z \in \mathbb{R}} \mathbb{E}_d\left[z + \frac{\max(Y^{(d)} - z, 0)}{1 - \beta}\right], \tag{17}$$

where $\beta \in [0, 1)$ determines the degree of risk sensitivity.

4) Exponential Weighted Average: Finally, we consider an Exponential Weighted Average strategy, which assigns smooth, non-linear importance to domain realizations based on their magnitude

$$\mathcal{A}_{\text{exp}}(\mathbf{Y}) = \sum_{d=1}^{D} w_d Y^{(d)}, \quad \text{with } w_d = \frac{\exp(\alpha Y^{(d)})}{\sum_{j=1}^{D}\exp(\alpha Y^{(j)})}, \tag{18}$$

where α is a temperature parameter controlling risk sensitivity.

IV. EXPERIMENTS

We evaluate our approach using the MuJoCo Playground [25], an open-source library for robot simulation, training, and sim-to-real transfer onto robots. By utilizing the standardized non-prehensile manipulation tasks provided within this ecosystem, we ensure our results are grounded in a reproducible and community-validated environment rather than a custom, hand-tuned scenario.

All experiments are conducted on a 7-DOF Franka Emika Panda robot equipped with a Robotiq 2F-85 gripper. We leverage the MJX backend [7] of MuJoCo [26] to execute massive, GPU-accelerated parallel rollouts for both the online SPC optimizer and the offline training of our baseline policies.

The robot is torque-controlled at the joint level, and the manipulated objects are modeled as 6-DOF free bodies. For all our experiments, common hyperparameters used are listed in Table I.

Our experiments are designed to investigate the following research questions:

1) **Performance:** Does Policy-guided CEM yield superior performance compared to the standalone nominal policy?
2) **Constraint Satisfaction:** Can Policy-guided CEM ensure constraints are satisfied?
3) **Geometric Generalization:** Does the RL policy provide a functional prior (e.g., "how to approach an object") that the MPC can adapt to unseen shapes in training?
4) **Parametric Robustness:** To what extent can the Policy-guided CEM compensate for significant distribution shifts in physical parameters (e.g., mass, friction)?

A. Task Description

The robot must push an object from a randomized initial position to a target pose specified by position and orientation. The gripper remains closed throughout the task, and manipulation is achieved through non-prehensile contact. The cost function for the non-prehensile push task is specified as

$$\ell(\mathbf{x}_t, \mathbf{u}_t) = w_p \|p_{\text{obj}} - p_{\text{tar}}\|^2 + w_g \|p_{\text{grip}} - p_{\text{app}}\|^2 + w_\theta \, e_\theta^2, \tag{19}$$

where $p_{\text{obj}}, p_{\text{tar}} \in \mathbb{R}^2$ are the object and target positions in the plane, $p_{\text{grip}} \in \mathbb{R}^3$ is the gripper position, p_{app} is a point offset from the object along the push direction, e_θ is the orientation error between object and target quaternions, $w_p, w_g, w_\theta > 0$ are weighting coefficients.

TABLE I: Experimental Hyperparameters

Parameter	Symbol	Value
CEM Controller		
Number of Samples	N	96
Elite Fraction	ζ	$\frac{8}{96}$
CEM Iterations	K	1
Initial Std. Dev.	σ_{start}	0.1
Planning Horizon	T	0.5 s
KL Divergence Weight	η	0.01
Cost Function		
Position Weight	w_p	10.0
Gripper Weight	w_g	5.0
Orientation Weight	w_θ	1.0

B. Experiment: Performance

To quantitatively evaluate the proposed method, we conduct 100 evaluation simulations in the environment across three control configurations:

1) POLICY: Pre-trained policy (baseline).
2) CEM: Standard Cross-Entropy Method.
3) POLICY-GUIDED CEM: Our proposed approach.

A trial is considered a *success* if the final position error is less than 0.05m and a final orientation error is less than 15°. Quantitative results are summarized in Table II. POLICY-GUIDED CEM outperforms both baselines across all metrics. Notably, the median time 2.3s to reach the success threshold is significantly faster than the pure CEM (4.45s) approach.

The mean position and orientation error trajectories over time are presented in Figure 2.a and 2.b respectively. The stand-alone POLICY exhibits a high steady-state error and the largest variance, often failing to precisely align the *Rectangular Prism*. While the CEM approach eventually converges to a lower error than the policy, its lack of a learned prior results in a "noisy" initial transient (see Figure 2.a between 0–2s) and slower overall convergence. In contrast, POLICY-GUIDED CEM benefits from the reactive strengths of the policy and the precision of online optimization. It exhibits the steepest descent in both position and orientation error, maintaining a tighter standard deviation throughout the task. This synergy allows POLICY-GUIDED CEM to achieve a 85.0% success rate, demonstrating a considerable performance improvement over the baseline policies 47.0% .

TABLE II: **Performance Comparison:** *Rectangular Prism*

	Policy	CEM	PG-CEM
Metric	Med. (IQR)	Med. (IQR)	Med. (IQR)
Final Pos. Err (m)	0.036 (0.02-0.08)	0.023 (0.01-0.09)	**0.017** (0.01-0.02)
Final Ori. Err (°)	11.3 (4.0-65.6)	5.9 (1.5-20.6)	**4.1** (2.1-7.3)
Total Cost	1112 (262-2594)	375 (158-1620)	**272** (151-847)
Time (s)	2.55 (1.8-4.9)	4.45 (2.8-10.0)	**2.30** (1.5-5.4)
Success Rate	47.0%	53.0%	**85.0%**

PG-CEM: Policy-guided CEM.

TABLE III: **Performance Comparison:** *Cube*

	Policy	CEM	PG-CEM
Metric	Med. (IQR)	Med. (IQR)	Med. (IQR)
Final Pos. Err (m)	0.040 (0.03-0.07)	**0.010** (0.01-0.02)	0.019 (0.01-0.03)
Final Ori. Err (°)	35.6 (9.2-97.4)	**0.9** (0.4-2.7)	2.1 (1.0-4.0)
Total Cost	827 (344-3429)	**182** (67-497)	193 (138-459)
Time (s)	**2.09** (1.2-3.6)	3.41 (1.8-6.3)	2.28 (1.1-3.5)
Success Rate	32.0%	87.0%	**99.0%**

PG-CEM: Policy-guided CEM.

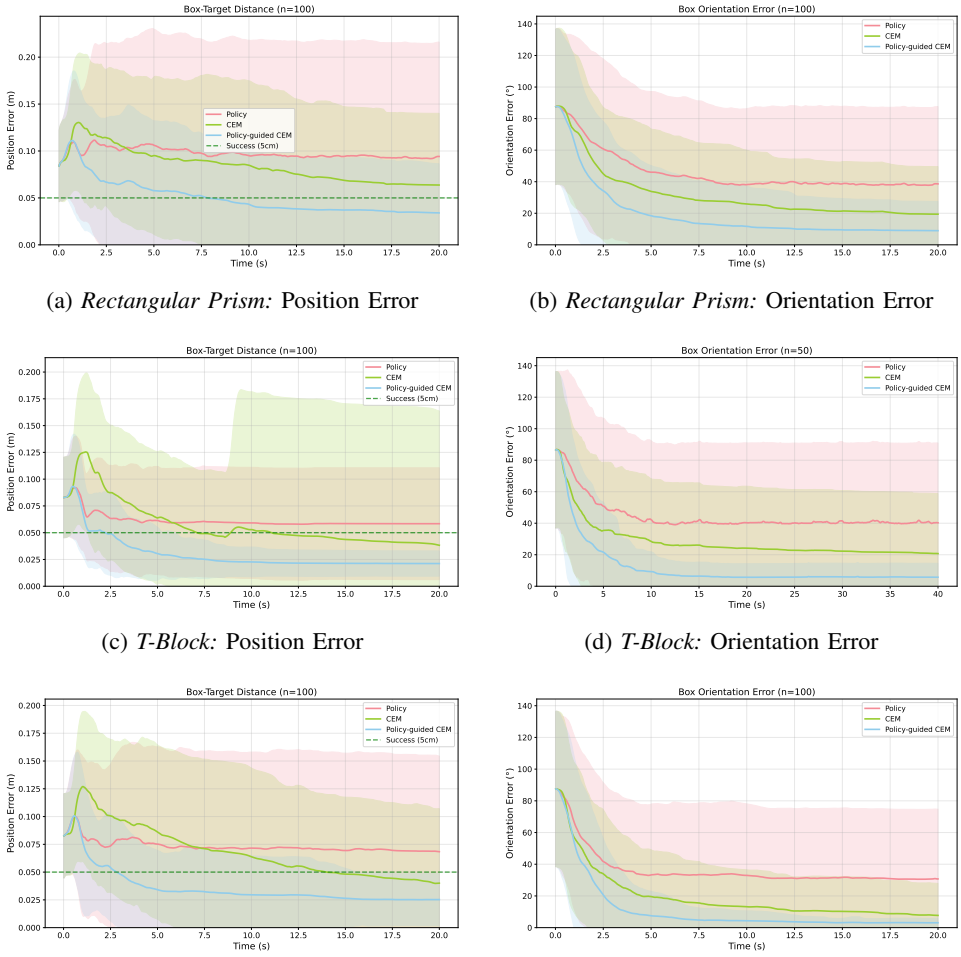
TABLE IV: **Performance Comparison:** *T-Block*

	Policy	CEM	PG-CEM
Metric	Med. (IQR)	Med. (IQR)	Med. (IQR)
Final Pos. Err (m)	0.029 (0.02-0.06)	**0.013** (0.01-0.03)	0.019 (0.01-0.03)
Final Ori. Err (°)	9.9 (4.5-23.1)	**1.4** (0.5-3.7)	2.1 (0.8-5.0)
Total Cost	326 (222-1621)	236 (100-756)	**187** (117-403)
Time (s)	1.92 (1.0-4.2)	4.77 (2.3-7.8)	**1.82** (1.0-3.2)
Success Rate	53.0%	79.0%	**93.0%**

PG-CEM: Policy-guided CEM.

C. Experiment: Geometric Generalization

To evaluate the zero-shot adaptability of our framework, we evaluated the performance on geometries not encountered during training. While the baseline RL policy was trained exclusively on a *Rectangular Prism*, we benchmarked the



(a) *Rectangular Prism:* Position Error

(b) *Rectangular Prism:* Orientation Error

(c) *T-Block:* Position Error

(d) *T-Block:* Orientation Error

(e) *Cube:* Position Error

(f) *Cube:* Orientation Error

Fig. 2: **Performance Comparison:** Mean metrics across 100 episodes for RL, CEM, and Policy-guided CEM. Shaded regions indicate one standard deviation.

POLICY, CEM, and POLICY-GUIDED CEM controllers across two additional shapes not seen in training (Figure 3):

1) *Cube*: An unseen symmetric object.
2) *T-Block*: An unseen asymmetric object with a more complex mass distribution.

These experiments aim to assess whether the RL policy establishes a functional behavioral prior, such as a fundamental "approach and push" motion, that an online optimizer can refine for objects with varying contact manifolds and inertial properties. Note, for these experiments, both CEM and POLICY-GUIDED CEM are provided with privileged knowledge of the object geometry during rollouts.

Performance results are summarized in Table III and IV for the *Cube* and *T-Block* respectively, and Figure 4 presents a visual comparison of the success rates. While the RL policy degrades significantly when faced with out-of-distribution *Cube* geometry, CEM alone is capable of leveraging its planning capability to achieve a high success rate of 87%. However, despite the baseline policy performing quite poorly in this scenario, the POLICY-GUIDED CEM is able to adapt the control online to achieve an improved success rate of 99%. Similarly, for the *T-Block* geometry, POLICY-GUIDED CEM is able to achieve a 40% success rate improvement over the baseline POLICY.

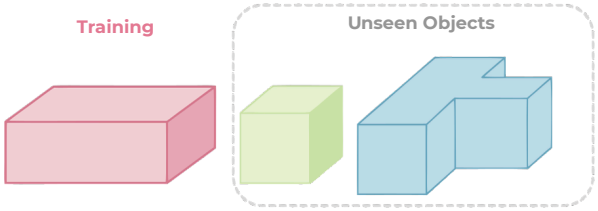


Fig. 3: *Rectangular Prism* (the nominal training geometry), *Cube*, and *T-Block* (unseen geometries during training).

D. Experiment: Parametric Robustness

While previous experiments provided the planner with privileged knowledge of the object geometry, practical applications often involve unknown or time-varying physical parameters. We therefore investigate whether POLICY-GUIDED CEM can exhibit robustness to dynamics mismatch without explicit uncertainty modeling or domain randomization within the planner.

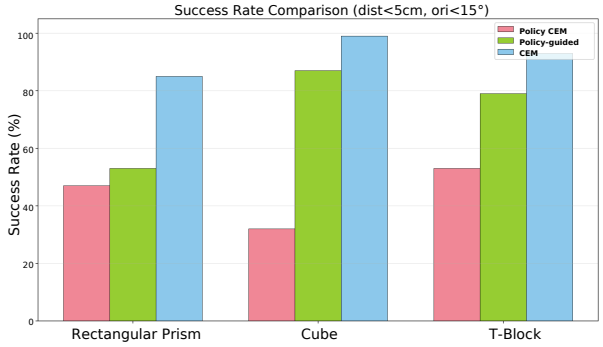


Fig. 4: **Geometric Generalization:** Success rate $[\%]$ (final distance error $<$ 5cm, final orientation error $<$ 15°) across three geometry variations over 100 evaluations.

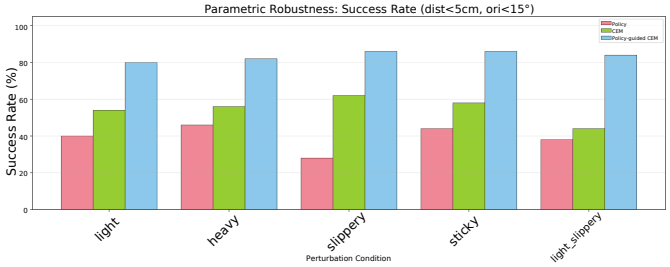


Fig. 5: **Parametric Robustness:** Success rate $[\%]$ across mass and friction variations over 100 evaluations.

We evaluate the controllers RL, CEM, and POLICY-GUIDED CEM controllers against the specific physical perturbations detailed in Table V, involving variations in mass and friction coefficients up to $2.0\times$ their nominal values. For this experiment, the internal dynamics model used by CEM and POLICY-GUIDED CEM remains fixed to the nominal physics parameters. The evaluation environment, however, is subject to significant perturbations.

As illustrated in Figure 5, POLICY-GUIDED CEM achieves consistently high success rates across all test conditions, demonstrating remarkable zero-shot transfer to perturbed dynamics. We hypothesize that this robustness stems from the stochastic nature of the CEM. As noted in recent literature regarding diffusion-inspired sampling-based control [27], the injection of noise into the action space, inherent to the sampling procedure, can improve performance under parametric uncertainty. By optimizing over a distribution of control inputs, the planner naturally favors trajectories that are robust to parameter perturbations.

E. Ablation Study: Online Domain Randomization

Given the inherent robustness observed in the previous experiment, we conducted an ablation study to investigate whether explicit Online Domain Randomization (DR) could further enhance performance.

In this setup, the POLICY-GUIDED CEM planner was modified to sample dynamics parameters during its internal rollouts. While the real environment remained fixed to a specific perturbation (e.g., *Heavy*, *Slippery*), the planner maintained

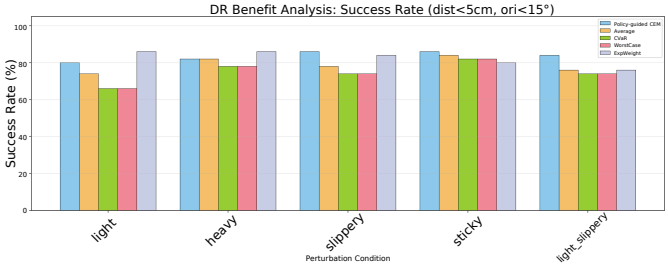


Fig. 6: **Online Domain Randomization:** Comparison of the baseline Policy-guided CEM (nominal model) against varying online domain randomization strategies.

uncertainty over the dynamics, sampling object mass from $\mathcal{U}(0.5, 2.0)\times$ nominal and friction from $\mathcal{U}(0.3, 2.0)\times$ nominal across 4 domains per iteration ($D = 4$). We compared the standard POLICY-GUIDED CEM (using the nominal model) against four risk strategies: AVERAGE, CVAR, WORSTCASE, and EXPWEIGHT (Exponential Weighting).

TABLE V: Experimental Perturbation Specifications

Condition	Mass Scale	Friction Scale
Light	$0.5\times$	$1.0\times$
Heavy	$2.0\times$	$1.0\times$
Slippery	$1.0\times$	$0.3\times$
Sticky	$1.0\times$	$2.0\times$
Light Slippery	$0.5\times$	$0.5\times$

The results are presented in Figure 6. Surprisingly, the inclusion of explicit domain randomization did not yield a statistically significant improvement over the baseline. The baseline POLICY-GUIDED CEM (Blue) performs comparably to, and often better than, the DR variants. While DR + EXPWEIGHT shows marginal gains in the *Light* and *Heavy* categories, it underperforms in *Slippery*, *Sticky* and *Light Slippery* conditions. The risk-averse strategies, particularly WORSTCASE and CVAR, generally resulted in lower success rates. This suggests that planning for the worst-case realization of dynamics in contact-rich tasks may lead to overly conservative behaviors that fail to exploit necessary contact modes. Additionally, implementing online DR requires simulating $D \times N$ parallel environments for every sample, increasing the computational memory footprint linearly with D.

These findings reveal that for our task, the input noise injected into the action space by the sampling distribution can provide sufficient robustness for a wide range of parametric uncertainties. Consequently, the standard POLICY-GUIDED CEM represents the most efficient point on the Pareto front of performance versus computational complexity. In future work, we wish to explore more difficult or varying tasks to assess more broadly how effective online domain randomisation is.

F. Experiment: Constraint Satisfaction

In real-world deployment, robots must operate within safe workspaces to avoid collisions or excursions into forbidden zones. We evaluate the capacity of our framework to strictly

enforce such state-space constraints. To tackle this, typical RL pipelines employ termination conditions paired with negative rewards (soft constraints) to discourage violations. The baseline Franka RL Policy used in this study was trained with a severe negative termination penalty for any object transition beyond the workspace boundary.

To test the limits of these approaches, we customized the Franka Push task to initialize goal object poses specifically near or slightly outside the workspace boundary, illustrated in Figure 7. We compare three control strategies:

1) POLICY: Standalone policy trained with soft penalties for constraint violations.
2) POLICY-GUIDED CEM: without constraint.
3) POLICY-GUIDED CEM: with constraint.

The constraint violation for the workspace is defined as:

$$c(\mathbf{x}) = \begin{cases} 1 & \text{if } p_{obj,x} > x_{max} \text{ or } p_{obj,x} < x_{min} \\ 1 & \text{if } p_{obj,y} > y_{max} \text{ or } p_{obj,y} < y_{min} \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

where $[p_{obj,x}, p_{obj,y}]^\top$ represents the Cartesian position of the center of the object in the xy-plane, and $\{x_{min}, x_{max}, y_{min}, y_{max}\}$ define the spatial boundaries.

Results are summarized in Table VI. The POLICY demonstrates the limitations of soft constraints, while it achieves a 0.00% violation rate (median $\%$ of steps where $c(\mathbf{x}) > 0$), it does so by becoming overly conservative, effectively failing to approach the target near the boundary, leading to a success rate of only 3.1%.

Conversely, the unconstrained POLICY-GUIDED CEM achieves the highest success rate (78.1%) and the lowest position error (0.03 m) by ignoring the boundaries, resulting in a massive 92.06% median violation rate. Our proposed POLICY-GUIDED CEM successfully unifies these objectives. It maintains the absolute safety of the base policy (0.00% violation) while leveraging online optimization to reach the target more effectively than the base policy. POLICY-GUIDED CEM achieves a 40.6% success rate, a $13\times$ improvement over the standalone RL policy, while strictly adhering to workspace constraints. This highlights the framework's ability to retain safety without sacrificing the performance gain obtained through online planning.

TABLE VI: **Constraint Satisfaction**: Performance Metrics

	Policy	PG-CEM	PG-CCEM
Metric	Med. (IQR)	Med. (IQR)	Med. (IQR)
Final Pos. Err (m)	0.05 (0.03-0.13)	**0.03** (0.02-0.03)	0.04 (0.03-0.07)
Final Ori. Err (°)	15.17 (5.29-25.83)	7.09 (4.48-11.22)	**4.72** (2.55-8.00)
Constr. Viol. (%)	**0.00** (0.00-0.00)	92.06 (29.47-96.92)	**0.00** (0.00-0.00)
Success Rate	3.1%	**78.1%**	40.6%

PG-CEM: without constraints. PG-CCEM: with constraints.

V. LIMITATIONS AND FUTURE WORK

Given the performance gains, robustness and zero-shot capabilities demonstrated in simulation, POLICY-GUIDED CEM is well-positioned for deployment and experimental validation on physical hardware. This will allow for a rigorous quantification of performance and an assessment of how effectively the approach can help bridge the sim-to-real gap.

Additionally, we aim to refine the optimization process by incorporating learned value functions directly into the MPC framework as a terminal cost $\phi(\mathbf{x}_T)$. While the policy serves as a control prior, the value function acts as a heuristic for the infinite-horizon "cost-to-go." By integrating this foresight, the controller can account for global task progress beyond the immediate planning horizon, which significantly improves stability and assists in avoiding local minima [28].

Finally, to further enhance the system's flexibility, we plan to transition from traditional physics-based simulators to learned latent dynamics models. By shifting planning into a learned latent space, the framework can more efficiently process high-dimensional sensory inputs and adapt to complex, non-analytical environmental effects such as soft-body deformations while simultaneously reducing the heavy dependency on high-fidelity physics engines.

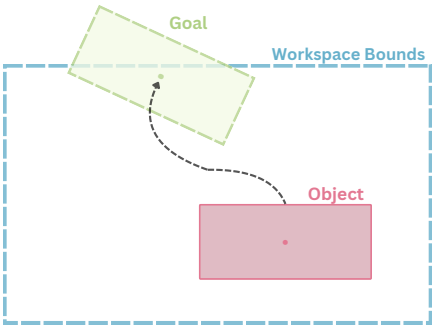


Fig. 7: **Constraint Satisfaction**: Task space visualization of the constrained Franka push experiment. The workspace boundary (blue) defines the region of valid object states. The target goal (green) is sampled near the boundary, requiring the optimizer to minimize pose error while strictly adhering to safety constraints.

VI. CONCLUSION

This paper presented Policy-guided CEM, a control framework that unifies the global task knowledge of reinforcement learning with the online planning capability of sampling-based predictive control. By integrating a pre-trained RL policy as an informative prior within a Cross-Entropy Method (CEM) optimizer, we leveraged the global task knowledge of reinforcement learning alongside the online adaptability and constraint-enforcement of model-based predictive control. We demonstrated that this synergy enables superior performance in contact-rich manipulation, exhibiting robust zero-shot generalization to unseen object geometries and significantly perturbed physical parameters. Future research will focus on physical hardware validation and the integration of learned value functions as terminal costs to provide improved long-horizon foresight.

REFERENCES

[1] M. T. Mason and K. M. Lynch, "Dynamic manipulation," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 1993, pp. 152–159.

[2] R. Deits and T. Koolen. (2023, Jan.) Picking up momentum. Boston Dynamics. [Online]. Available: https://bostondynamics.com/blog/picking-up-momentum/

[3] A. H. Li, P. Culbertson, V. Kurtz, and A. D. Ames, "DROP: Dexterous reorientation via online planning," arXiv:2409.14562, 2024.

[4] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1433–1440.

[5] T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa, "Predictive sampling: Real-time behaviour synthesis with mujoco," arXiv:2212.00541, 2022.

[6] R. Rubinstein, "The cross-entropy method for combinatorial and continuous optimization," *Methodology and Computing in Applied Probability*, vol. 1, pp. 127–190, 1999.

[7] MuJoCo XLA Authors, "Mujoco xla (mjx) user guide," https://mujoco.readthedocs.io/en/stable/mjx.html, 2025.

[8] Genesis Authors, "Genesis: A universal and generative physics engine for robotics and beyond," https://github.com/Genesis-Embodied-AI/Genesis, Dec. 2024.

[9] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," arXiv:2108.10470, 2021.

[10] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[11] V. Kurtz, A. Castro, A. O. Önol, and H. Lin, "Inverse dynamics trajectory optimization for contact-implicit model predictive control," arXiv:2309.01813, 2023.

[12] T. Erez and E. Todorov, "Trajectory optimization for domains with contacts using inverse dynamics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 4914–4919.

[13] A. Aydinoglu, A. Wei, W.-C. Huang, and M. Posa, "Consensus complementarity control for multi-contact mpc," *IEEE Transactions on Robotics*, 2024.

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

[15] Q. Le Lidec, F. Schramm, L. Montaut, C. Schmid, I. Laptev, and J. Carpentier, "Leveraging randomized smoothing for optimal control of nonsmooth dynamical systems," *Nonlinear Analysis: Hybrid Systems*, vol. 52, p. 101468, 2024.

[16] A. Jordana, S. Kleff, A. Haffemayer, J. Ortiz-Haro, J. Carpentier, N. Mansard, and L. Righetti, "Infinite-horizon value function approximation for model predictive control," *IEEE Robotics and Automation Letters*, vol. 10, no. 7, pp. 7563–7570, 2025.

[17] S. H. Jeon, H. J. Lee, S. Hong, and S. Kim, "Residual mpc: Blending reinforcement learning with gpu-parallelized model predictive control," 2025. [Online]. Available: https://arxiv.org/abs/2510.12717

[18] P. Wang, C. Li, C. Weaver, K. Kawamoto, M. Tomizuka, C. Tang, and W. Zhan, "Residual-mppi: Online policy customization for continuous control," 2025. [Online]. Available: https://arxiv.org/abs/2407.00898

[19] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. Del Prete, "Optimization-based control for dynamic legged robots," *IEEE Transactions on Robotics*, 2023.

[20] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

[21] T. O'Brien, "Sampling-based predictive control for non-prehensile manipulation," *Australasian Conference on Robotics and Automation (ACRA 2025)*, 2025.

[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[23] Z. Liu, H. Zhou, B. Chen, S. Zhong, M. Hebert, and D. Zhao, "Constrained Model-based Reinforcement Learning with Robust Cross-Entropy Method," *arXiv e-prints*, p. arXiv:2010.07968, Oct. 2020.

[24] V. Kurtz and J. W. Burdick, "Generative predictive control: Flow matching policies for dynamic and difficult-to-demonstrate tasks," 2025. [Online]. Available: https://arxiv.org/abs/2502.13406

[25] K. Zakka, B. Tabanpour, Q. Liao, M. Haiderbhai, S. Holt, J. Y. Luo, A. Allshire, E. Frey, K. Sreenath, L. A. Kahrs, C. Sferrazza, Y. Tassa, and P. Abbeel, "Mujoco playground," 2025. [Online]. Available: https://arxiv.org/abs/2502.08844

[26] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 5026–5033.

[27] H. Xue, C. Pan, Z. Yi, G. Qu, and G. Shi, "Full-order sampling-based mpc for torque-level locomotion control via diffusion-style annealing," arXiv:2409.15610, 2024.

[28] A. Jordana, S. Kleff, A. Haffemayer, J. Ortiz-Haro, J. Carpentier, N. Mansard, and L. Righetti, "Infinite-horizon value function approximation for model predictive control," *IEEE Robotics and Automation Letters*, 2025, accepted June 2025.