Robust Sampling-Based Control of Mobile Manipulators for Interaction With Articulated Objects

Giuseppe Rizzi, Jen Jen Chung, *Member, IEEE*, Abel Gawel, Lionel Ott, Marco Tognon, *Member, IEEE*, and Roland Siegwart, *Fellow, IEEE*

***Abstract*—In this article, we investigate and deploy sampling-based control techniques for the challenging task of the mobile manipulation of articulated objects. By their nature, manipulation tasks necessitate environment interactions, which require the handling of nondifferentiable switching contact dynamics. These dynamics represent a strong limitation for traditional gradient-based optimization methods, such as model-predictive control and differential dynamic programming, which often rely on heuristics for trajectory generation. *Sampling-based* techniques alleviate these constraints but do not ensure robots' stability and input/state constraints either. On the other hand, real-world applications in human environments require safety and robustness to unexpected events. For this reason, we propose a novel framework for safe robotic manipulation of movable articulated objects. The framework combines sampling-based control together with *control barrier functions* and *passivity theory* that, thanks to formal stability guarantees, enhance the safety and robustness of the method. We also provide the practical insights that enable robust deployment of stochastic control using a conventional central processing unit. We deploy the algorithm on a ten-degree-of-freedom mobile manipulator robot. Finally, we open source our generic and multithreaded implementation.**

***Index Terms*—Manipulation of articulated objects, mobile manipulation, motion control of manipulators, optimization and optimal control.**

I. Introduction

THERE is a growing interest in deploying autonomous systems in unstructured environments to perform complex manipulation tasks for different applications like assistive service in the health-care domain [1], agrifoods [2], and industrial inspection and maintenance [3] . In those scenarios, the community is particularly interested in the manipulation of movable and articulated objects (like the furniture in Fig. 1), which poses additional challenges beyond those experienced in a static environment. Mobile manipulator robots present a compelling choice to tackle these problems since they combine an unconstrained workspace with highly dexterous interaction capabilities. However, to fully exploit these capabilities, systems require planning and control algorithms that can generate fast, accurate, and coordinated reactive whole-body motions that account for multiple potential contacts with the environment.

Manuscript received 4 May 2022; revised 13 October 2022; accepted 12 December 2022. This work was supported by the European Union H2020 Program under project PILOTING H2020-ICT-2019-2 871542 and project HERON under Grant 955356. This paper was recommended for publication by Associate Editor A. Dietrich and Editor E. Yoshida upon evaluation of the reviewers' comments. *(Corresponding author: Giuseppe Rizzi.)*

Giuseppe Rizzi, Lionel Ott, and Roland Siegwart are with the Autonomous Systems Lab, ETH Zürich, 8092 Zürich, Switzerland (e-mail: grizzi@ethz.ch; lionel.ott@mavt.ethz.ch; rsiegwart@ethz.ch).

Jen Jen Chung is with the School of Information Technology and Electrical Engineering, the University of Queensland, Brisbane, QLD 4072, Australia (e-mail: jenjen.chung@uq.edu.au).

Abel Gawel is with the Huawei Technologies, 8050 Zürich, Switzerland (e-mail: abel.gawel@googlemail.com).

Marco Tognon was with the Autonomous Systems Lab, ETH Zürich, 8092 Zürich, Switzerland. He is now with INRIA Rennes, 35042 Rennes, France (e-mail: marco.tognon@inria.fr).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TRO.2022.3233343.

Digital Object Identifier 10.1109/TRO.2022.3233343

A. Related Works

While traditional "plan-and-act" frameworks break down manipulation tasks into subproblems that are easier to solve (e.g., reach, grasp, and pull) [4], they do not offer fast replanning, which is crucial for mobile manipulation in dynamic and uncertain environments.

With the recent advancements in artificial intelligence, reinforcement learning (RL) is a promising method to solve a range of robotic control tasks, including manipulation [5], as they learn an end-to-end representation of the optimal policy. However, real-world applications of RL typically require training times that are not practical for physical hardware and suffer from the well-known *sim-to-real* gap [6]. On the other side of the spectrum, model-predictive control (MPC) has gained broad interest in the robotics community, thanks to its ability to deal with input constraints and task objectives by solving a multivariate optimization problem or using the *principle of optimality*. MPC has been successfully applied to aerial robots [7], autonomous racing [8], legged locomotion [9], and whole-body control [10]. Nevertheless, MPC requires a model that is locally differentiable with respect to the input and the state [11]. On the other hand, manipulation tasks involve changes in the contact state causing sharp discontinuities in both the cost and system dynamics, thus directly violating the differentiability requirements. Recently, sampling-based methods have emerged and advanced in theory and applications [12], [13], [14], [15], [16]. In contrast to traditional MPC, sampling methods stem from a probabilistic interpretation of the control problem. Rather than solving a complex optimization problem, they rely on sampling system trajectories, referred to as *rollouts*, and "weighting" them according to the cumulative cost so that only favorable

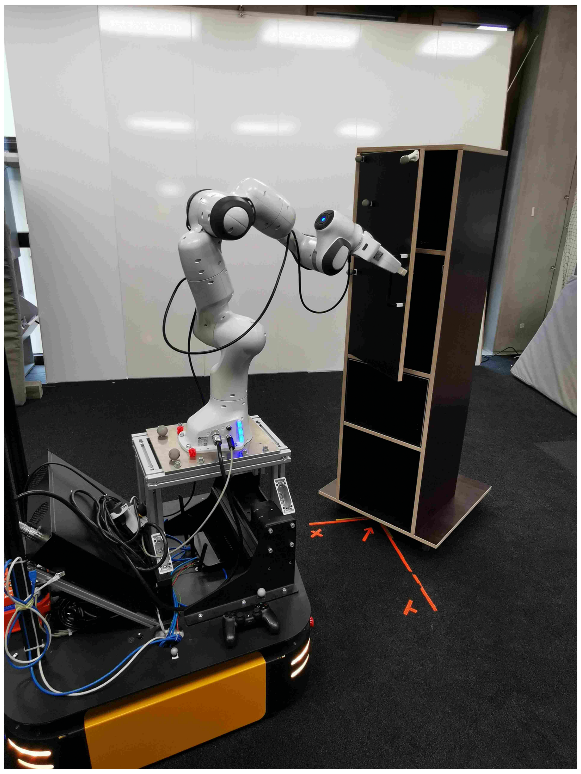


Fig. 1. *RoyalPanda* (a ten-DOF mobile manipulator) while performing a door opening task.

trajectories survive the iterative sampling process. The only requirement is that it is possible to forward simulate the system evolution. This has been exploited to control camera motions for target tracking in drone racing [12], robot arm motions for manipulation tasks [13], and for generating aggressive driving maneuvers such as drifting [14], [15].

Demonstrations on real systems involving different physical interactions (e.g., a robot arm opening a drawer [13]) have typically only been shown by breaking down the multicontact task into stages and enforcing constraints when switching between them. Often these methods stem from a first principle analysis of the problem and rely on strong simplifications, heuristics, or precomputed interaction references. They often follow the established control framework combining a high-level heuristic-based reference generator and a low-level tracking controller [9], [17], [18]. In this context, MPC is often used to track an externally provided reference trajectory. In [10], for example, a circular trajectory is computed for a door opening task, and the gripper is rigidly connected to the handle *prior* to the interaction. This can limit the control envelope of the system and sacrifices solution optimality. For example, as in [19], it is common practice to fix the gripper orientation between successive reach and pull stages and perform manipulation under a rigid grasp. In the presence of uncertainty and tracking errors, this often leads to high contact forces and dangerous behaviors. Even replanning might fail since heuristics do not scale to most complex robot configurations. For those, it is often unintuitive to think of the type of interaction that achieves the task. In contrast, our sampling-based controller is deployed as an interaction-aware and closed-loop reference generator that can fully exploit the contact dynamics. We have observed that the controller was able to use different contact points on the robot hand to interact with the door, not just the fingers' tips. Furthermore, we deploy a *hook* single finger design for nonprehensile manipulation, alleviating the constraint of a rigid grasp.

In order to avoid safety-critical configurations, additional cost terms are often formulated. However, while these penalize unsafe paths in sampling-based methods, they do not prohibit them. As a consequence, constraint fulfillment merely relies on sampling "safe" trajectories. Recently, *control barrier functions* (CBFs) have been introduced as a mean to constrain the system to a safe set. The safe set defines the locus where all safety requirements are met, e.g., no self-collision happens and joint limits are fulfilled. CBFs are expressed as affine inequality constraints in the control input that, when satisfied pointwise in the candidate safe set, imply forward invariance of the set and hence safety [20]. This control method has been successfully deployed in safety-critical applications such as aerial physical interaction [21], adaptive cruise control and lane keeping [22], segway stabilization [23], and human–robot collaboration [24]. CBFs have also been recently combined with an RL agent to provide safety while deploying a learnt policy. The work in [25], for example, learns barrier and Lyapunov functions, which are robust to model uncertainties. The task is achieved by a control Lyapunov function, which tracks a precomputed swing trajectory for a simulated planar bipedal robot. In [26], CBFs are instead used to guide the learning process. Rather than bootstrapping a new policy update, which may operate in an unsafe or irrelevant area of the state space, they update the policy around the previously deployed controller containing all previous CBF contributions. Taking inspiration from previous works, we similarly try to combine CBFs with sampling-based control methods for the reactive generation of interaction behaviors. In addition, we deploy CBFs to also guide the sampling of input trajectories in safe areas, complementing for poor sampling performance near the constraints.

Last but not least, sampling-based methods do not formally guarantee stability, which, especially during autonomous interaction, is as important as performance. In this regard, we are interested in a stable interaction with an *a priori* poorly known environment. This lack of knowledge might come from sensing limitations or model mismatches. *Passivity theory* [27] has drawn attention in recent years as a way to analyze the stability of a controlled system. Intuitively, a passive system cannot produce more energy than what it is provided with. A rather new approach to ensure *passivity* is to use an *energy tank*, i.e., an auxiliary virtual system that serves as a storing element containing the maximum energy available to perform the task [28]. When more than this energy budget is demanded, actions that could destabilize the system are prohibited, and the system is made passive. Energy tanks have been successfully deployed in many robotics applications such as for impedance controllers with time-varying stiffness [29], force–impedance control [30], [31], and lately with CBFs for controlling a robotic arm [24].

B. Contributions

The novelty of the proposed method is in bringing together sampling-based control, barrier methods, and passivity theory. Sampling-based control is able to find contact-rich reference trajectories, while barrier functions and passivity theory add a safety layer that accounts for kinematic and dynamics constraints. The safety and stability of the system are ensured through a sequential optimization problem that finds the closest policy to the sampled one.

In summary, the main contributions of this article are: 1) combining for the first time sampling-based control, barrier methods, and passivity theory in a way that each component complements the others' limitations; 2) providing practical insights that enhance the performance of the presented algorithm and guide practitioners for successful hardware transfer; and 3) open sourcing an efficient C++ implementation of the proposed method including a multithreaded sampling-based controller and an efficient quadratic program for addressing kinematic and dynamics constraints.

To demonstrate the applicability and effectiveness of this approach, we perform several numerical and experimental studies where we deploy the algorithms on our *RoyalPanda* platform (a seven-degree-of-freedom (DOF) Franka Emika Panda arm mounted on the holonomic Clearpath Ridgeback base, shown in Fig. 1) for a target reaching and door opening task. A video of the experiments can be found at https://youtu.be/Q1AohXmsLs4. An open-source implementation of our solution is provided at https://git.io/JXi4d.

To the best of the authors' knowledge, this framework has never been applied to whole-body control and manipulation of articulated objects.

C. Overview

The rest of this article is organized as follows. In Section II, the manipulation problem is formulated. In Section III, we review the main theoretical background. These theoretical tools are combined in a unified control method in Section IV. We summarize the main practical insights and aspects of the algorithms in Section V. We then evaluate the overall method in Section VI by simulation and hardware experiments. We present a qualitative discussion of the method's limitations and future research directions in Section VII. Finally, Section VIII concludes this article.

II. Modeling and Problem Formulation

In this article, we consider the challenging task of manipulating articulated objects[1] with a mobile manipulator through *nonprehensile manipulation*. We define $\boldsymbol{q} \in \mathbb{R}^{n_q}$ and $\dot{\boldsymbol{q}} \in \mathbb{R}^{n_q}$ as the vectors of the robot configuration and its time derivative, respectively, where $n_q \in \mathbb{N}_{>0}$ is the robot's DOF. Similarly, we describe the object configuration and corresponding time derivative by $\boldsymbol{o} \in \mathbb{R}^{n_o}$ and $\dot{\boldsymbol{o}} \in \mathbb{R}^{n_o}$, respectively, where $n_o \in \mathbb{N}_{>0}$ is the object's DOF. The state vector is defined by

$$\boldsymbol{x} = \begin{bmatrix}\boldsymbol{q}^\top & \dot{\boldsymbol{q}}^\top & \boldsymbol{o}^\top & \dot{\boldsymbol{o}}^\top\end{bmatrix}^\top \in \mathbb{R}^{2(n_q+n_o)}. \quad (1)$$

The time evolution of the state is given by $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$, where

$$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{M}_r^{-1}\left(\boldsymbol{J}_r^\top \boldsymbol{f}_{\text{ext}} - \boldsymbol{C}_r(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{g}_r(\boldsymbol{q}) + \boldsymbol{\tau}_{\text{cmd}}(\boldsymbol{u})\right) \\ \dot{\boldsymbol{o}} \\ \boldsymbol{M}_o^{-1}\left(-\boldsymbol{J}_o^\top \boldsymbol{f}_{\text{ext}} - \boldsymbol{C}_o(\boldsymbol{o},\dot{\boldsymbol{o}})\dot{\boldsymbol{o}} - \boldsymbol{g}_o(\boldsymbol{o})\right) \end{bmatrix}. \quad (2)$$

$\boldsymbol{M}_r \in \mathbb{R}^{n_q \times n_q}$ and $\boldsymbol{M}_o \in \mathbb{R}^{n_o \times n_o}$ represent the inertia matrices, while $\boldsymbol{J}_r(\boldsymbol{q}) \in \mathbb{R}^{n_q \times 3}$ and $\boldsymbol{J}_o(\boldsymbol{o}) \in \mathbb{R}^{n_o \times 3}$ are the Jacobians at the robot and object contact point,[2] respectively. $\boldsymbol{J}_r^T$ and $\boldsymbol{J}_o^T$ map the interaction force $\boldsymbol{f}_{\text{ext}} \in \mathbb{R}^3$ at the contact point into the efforts at the object and robot joints. Robot and object Coriolis and gravity terms are denoted as $\boldsymbol{C}_r(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$, $\boldsymbol{C}_o(\boldsymbol{o},\dot{\boldsymbol{o}})\dot{\boldsymbol{o}}$ and $\boldsymbol{g}_r(\boldsymbol{q})$, $\boldsymbol{g}_o(\boldsymbol{o})$, respectively.

The system input $\boldsymbol{u} \in \mathbb{R}^{n_q}$ are the desired robot joint velocities $\dot{\boldsymbol{q}}^*$. The joint torques, denoted by $\boldsymbol{\tau}_{\text{cmd}} \in \mathbb{R}^{n_q}$, are computed by a low-level velocity controller as a function of the velocity references $\boldsymbol{u}$. We define with $\mathcal{X} \subseteq \mathbb{R}^{2(n_q+n_o)}$ and $\mathcal{U} \subseteq \mathbb{R}^{n_q}$ as the spaces of admissible states and inputs, respectively.

The control trajectory is defined as a sequence of control inputs over a time horizon T and starting at time t: $U_t = \{\boldsymbol{u}_t, \boldsymbol{u}_{t+\Delta t}, \ldots, \boldsymbol{u}_{t+T-\Delta t}\}$ with Δt the time step. Each command sequence is sampled from a feedback policy distribution $U_t \sim \pi_{\boldsymbol{\theta}}$, where $\boldsymbol{\theta}$ are the distribution parameters. We denote with $X_t = \{\boldsymbol{x}_t, \boldsymbol{x}_{t+\Delta t}, \ldots, \boldsymbol{x}_{t+T}\}$ the state sequence obtained by rolling out a policy sample U_t when starting at the current state $\boldsymbol{x}_t$.

The control objective is to find the feedback policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{x}_t)$ that minimizes some statistics over an objective metric $l : \mathcal{X} \times \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}$ that maps for each time t, the system state $\boldsymbol{x}_t$, the desired state $\boldsymbol{x}_t^*$ and command $\boldsymbol{u}_t$ to a scalar value. We can now formulate the control objective as an optimization problem

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & \mathbb{E}_{\pi\theta} \int_t^\infty l(\boldsymbol{x}_t^*, \boldsymbol{x}_t, \boldsymbol{u}_t)\, dt \\ \text{s.t.} \quad & \dot{\boldsymbol{x}}_t = f(\boldsymbol{x}_t, \boldsymbol{u}_t) \quad \forall\, t \\ & \boldsymbol{u}_t \sim \pi_{\boldsymbol{\theta}}(\boldsymbol{x}_t) \quad \forall\, t \\ & \boldsymbol{x}_t \in \mathcal{X} \quad \forall\, t \\ & \boldsymbol{u}_t \in \mathcal{U} \quad \forall\, t \end{aligned} \quad (3)$$

Commonly, the objective in (3) is simplified to the sum of a finite-horizon and final cost term that approximates the infinite horizon. In the discrete-time setting, we have

$$\mathop{\mathbb{E}}_{\pi_{\boldsymbol{\theta}}}\left[\int_t^\infty l(\boldsymbol{x}^*, \boldsymbol{x}_t, \boldsymbol{u}_t) dt\right]$$

[1]We define articulated objects as nonactuated objects composed of more than one rigid part connected by joints allowing rotations or translations.

[2]Without loss of generality, we consider single contacts to simplify the notation. Nevertheless, extension to the multicontact case is straightforward.

$$\approx \underbrace{\mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[c_{\text{term}}(\boldsymbol{x}_{t+T}) + \sum_{k=0}^{K} c(\boldsymbol{x}_{t+k\Delta t}, \boldsymbol{u}_{t+k\Delta t})\right]}_{J(X_t,U_t)}. \tag{4}$$

For the sake of notation simplicity, we have omitted the dependence from the desired state $\boldsymbol{x}_t^*$. The cost function $c(\boldsymbol{x}, \boldsymbol{u}) \in \mathbb{R}_{\geq 0}$ maps the current state and input into a nonnegative scalar, which indicates how close the state and commands are to the goal, t and $t+T$ are the initial and final time of the horizon, K is the number of discrete steps, and $c_{\text{term}}(\boldsymbol{x}_{t+T}) \in \mathbb{R}_{\geq 0}$ is the terminal cost that approximates the tail of the infinite-horizon cumulative cost, which is denoted as $J(X_t, U_t)$.

III. Preliminaries

As the proposed method uses a combination of theoretical tools that span different applications and previous works, we present a short summary that can help the reader to better understand the rest of this article and build some preliminary background.

A. Sampling-Based Control

In contrast to deterministic policy optimization, we look at the optimal control problem (3) from a Bayesian perspective. This approach has many similarities with policy gradient methods used in RL [32]. The key difference is that the method is used *online* to iteratively update a parametric policy. While there are several avenues to derive the update equation, for example, *variational inference* [33] or *free energy* [15], here, we will refer to *stochastic control* and follow a Bayesian approach similar to [34].

The control problem is formulated introducing an additional binary random variable $\mathcal{O}$ whose value is 1 when a trajectory is optimal and 0 when it is not. The optimal control objective is to find the optimal input parameters $\boldsymbol{\theta}$, which maximize the success probability $p(\mathcal{O}=1)$. In order to improve convergence, we optimize its logarithm, as it has high gradients in the domain where the probability is low

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \log p(\mathcal{O}=1). \tag{5}$$

The success likelihood can be further expanded to make explicit its dependence on the policy parameters

$$\begin{aligned} p(\mathcal{O}=1) &= \int_{\pi_{\boldsymbol{\theta}}} p(\mathcal{O}|X_t, U_t; \boldsymbol{x}_t) p(X_t, U_t; \boldsymbol{x}_t) && (6)\\ &= \int_{\pi_{\boldsymbol{\theta}}} p(\mathcal{O}|X_t, U_t) p(X_t|U_t) p(U_t) && (7)\\ &= \int_{\pi_{\boldsymbol{\theta}}} p(\mathcal{O}|X_t, U_t) \pi_{\boldsymbol{\theta}}(U_t) && (8)\\ &= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[p(\mathcal{O}|X_t, U_t)]. && (9)\end{aligned}$$

The step in (7) follows from the assumption of deterministic dynamics, and we drop the explicit dependence on $\boldsymbol{x}_t$ as this is already contained in the state sequence X_t. Any monotonically decreasing function $g(\cdot): \mathbb{R} \to \mathbb{R}_{>0}$ can be used to map costs to a *pseudo success likelihood*. Often the exponential function is used as the success likelihood function

$$\begin{aligned} \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[p(\mathcal{O}|X_t, U_t)] &\propto \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\mathcal{J}(X_t, U_t)] && (10)\\ &= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\exp(-\lambda J(X_t, U_t))] && (11)\end{aligned}$$

where the higher the λ, the lower the cost needs to be in order to be mapped to a small *success likelihood*. To optimize the objective in (5) we perform gradient descent

$$\boldsymbol{\theta}^{i+1} = \boldsymbol{\theta}^i + \rho \nabla_{\boldsymbol{\theta}} \log \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\mathcal{J}(X_t, U_t)]. \tag{12}$$

We choose the exponential function to map costs to likelihoods and model the policy with a Gaussian distribution, which is parametric in the mean $U_t \sim \{\mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}), \mathcal{N}(\boldsymbol{\mu}_{t+\Delta t}, \boldsymbol{\Sigma}), \ldots, \mathcal{N}(\boldsymbol{\mu}_{t+T-\Delta t}, \boldsymbol{\Sigma})\} = \pi_{\boldsymbol{\theta}}$, where $\boldsymbol{\theta} = \{\boldsymbol{\mu}_t, \boldsymbol{\mu}_{t+\Delta t}, \ldots, \boldsymbol{\mu}_{t+T-\Delta t}\}$ are the mean vectors around which the input is sampled at each time step. $\boldsymbol{\Sigma} \in \mathbb{R}^{n_u \times n_u}$ is the diagonal covariance matrix of the multinomial distribution. The derivation of the gradient step equation for this particular design choice can be found in Appendix A. In the end, we obtain the following update equation for the kth mean vector:

$$\boldsymbol{\mu}_k^{i+1} = \boldsymbol{\mu}_k^i + \rho \boldsymbol{\Sigma}^{-1} \frac{\mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\exp(-\lambda J)\boldsymbol{\varepsilon}_k]}{\mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\exp(-\lambda J)]} \tag{13}$$

where $\boldsymbol{\varepsilon}_k = \boldsymbol{u}_k - \boldsymbol{\mu}_k$ is the random policy perturbation at time k. The step size can be decoupled from the noise variance: $\rho = \alpha \boldsymbol{\Sigma}$. Finally, the expectation can be estimated empirically via Monte Carlo sampling, obtaining the final policy update rule

$$\boldsymbol{\mu}_k^{i+1} = \boldsymbol{\mu}_k^i + \alpha \sum_{l=0}^{N-1} \omega_l \boldsymbol{\varepsilon}_k^l \tag{14}$$

$$\omega_l \approx \frac{\exp(-\lambda J_l)}{\sum_{s=0}^{N-1} \exp(-\lambda J_s)} \tag{15}$$

where N is the number of sampled rollouts. The input sequence applied to the system is then chosen as the *maximum-likelihood estimator* of the policy. This is given by the mean of each Gaussian, modeling the input distribution for each time step: $U_t^* = \{\boldsymbol{\mu}_t, \ldots \boldsymbol{\mu}_{t+T-\Delta t}\} = \boldsymbol{\theta}$.

B. Control Barrier Functions

It is desirable to design a controller that mediates performance and safety. *CBFs* have been recently introduced as a means to unify these objectives. This theory is based on the concept of *forward invariance* of a safe subset $\mathcal{C}$ of the state space, which is where safety conditions are met. A set $\mathcal{S}$ is called *forward invariant* if for every $\boldsymbol{x}_0 \in \mathcal{S}$, $\boldsymbol{x}(t) \in \mathcal{S}$ for every t. A differentiable function $h(\boldsymbol{x})$ characterizes the safe set, such that

$$\mathcal{C} = \{\boldsymbol{x} \in \mathbb{R}^n : h(\boldsymbol{x}) \geq 0\} \tag{16}$$

$$\partial\mathcal{C} = \{\boldsymbol{x} \in \mathbb{R}^n : h(\boldsymbol{x}) = 0\} \tag{17}$$

$$\text{Int}(\mathcal{C}) = \{\boldsymbol{x} \in \mathbb{R}^n : h(\boldsymbol{x}) > 0\} \tag{18}$$

where $\partial\mathcal{C}$ and $\text{Int}(\mathcal{C})$ denote the boundary and interior of the safe set, respectively. The function $h(\boldsymbol{x})$ is a *zeroing barrier* *function* (ZBF) for the set $\mathcal{C}$ if there exist a $\gamma > 0$ and a set $\mathcal{D}$ with $\mathcal{C} \subseteq \mathcal{D} \subset \mathbb{R}^n$ such that, $\forall \boldsymbol{x} \ \in \mathcal{D}$

$$\dot{h}(\boldsymbol{x}) \geq -\gamma h(\boldsymbol{x}). \tag{19}$$

The existence of a ZBF implies the forward invariance of $\mathcal{C}$ [20]. Furthermore, defining the ZBF on a larger set than $\mathcal{C}$ allows the system to be more robust to model perturbations.[3] The condition in (19) can be rewritten for a controlled affine system as

$$\sup_{\boldsymbol{u}\in\mathcal{U}} \left[\frac{\partial h}{\partial \boldsymbol{x}}(\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u}) + \gamma h(\boldsymbol{x})\right] \geq 0 \quad \forall \boldsymbol{x} \in \mathcal{D}. \tag{20}$$

When this inequality is fulfilled by the control input, the system will be made forward invariant. As the constraint is affine in the control input, it can be solved via a quadratic optimization problem (QP). The advantage of a QP is that it allows trading-off performance and safety through ZBF. When a feedforward control is available from a nominal controller, a minimum perturbation on the feedforward command vector $\boldsymbol{u}_{\text{ff}}$ can be found through the following QP [35]:

$$\begin{aligned} \arg\min_{\boldsymbol{u}\in\mathbb{R}^n} \quad & \tfrac{1}{2}||\boldsymbol{u} - \boldsymbol{u}_{\text{ff}}|| \\ \text{s.t.} \quad & \dot{h}(\boldsymbol{x}) \geq -\gamma h(\boldsymbol{x}). \end{aligned} \tag{21}$$

In practice, the control input is constrained to the feasible set, and therefore, controlled invariance cannot generally be guaranteed for the real system. As a remedy, we add input limits as hard constraints while softening CBF constraints with the use of slack variables.[4]

C. Passivity and Energy Tank

It is important that the autonomous system preserves passive behavior during interaction, which in turn implies stability. Consider a system with state $\boldsymbol{x} \in \mathbb{R}^n$, input $\boldsymbol{u} \in \mathbb{R}^m$, and output $\boldsymbol{y} \in \mathbb{R}^m$. This system is said to be *passive* w.r.t $(\boldsymbol{u}, \boldsymbol{y})$ if for all inputs and initial states, there exists a positive-semidefinite *storage function* $S : \mathbb{R}^n \to \mathbb{R}_+$ such that for any time t

$$S(\boldsymbol{x}(t)) - S(\boldsymbol{x}(0)) \leq \int_0^t \boldsymbol{u}^T \boldsymbol{y}\, d\tau. \tag{22}$$

In other words, no additional energy can be produced other than what is flowing into the system through the *power port* $(\boldsymbol{u}, \boldsymbol{y})$. For an autonomously controlled system (including the environment a robot is interacting with), this condition translates to

$$\dot{S} \leq P_{\text{in}} = 0 \tag{23}$$

where P_{in} is the power flowing into the system. As shown in [30], autonomous passivity can be enforced by interconnecting the controlled system with a secondary passive system, called the *energy tank*, whose energy is bounded. The energy tank has state $x_t \in \mathbb{R}$ and storage function $S(t) = \frac{1}{2}x_t^2 \in \mathbb{R}$, with $S_t(0)$ as its initial value. Its time evolution is described by

$$\begin{cases} \dot{x}_t = u_t(t) \\ y_t(t) = \frac{\partial S}{\partial x_t} = x_t(t) \end{cases} \tag{24}$$

where $(u_t(t), y_t(t))$ is the power port through which the tank can exchange energy with the interconnected system. By interconnecting the controlled system with the tank, a passivity-violating control action will result in the depletion of the tank, restoring the passive balance of power flows. When the tank has no energy left, nonpassive behaviors cannot be implemented anymore. The interconnection can be implemented as follows:

$$\begin{cases} u_t(t) = \boldsymbol{a}^T(t)\boldsymbol{u} \\ \boldsymbol{y} = \boldsymbol{a}(t)y_t(t) \end{cases} \tag{25}$$

where $\boldsymbol{a}(t) \in \mathbb{R}^m$ is defined as

$$\boldsymbol{a}(t) = \frac{\boldsymbol{\gamma}(t)}{x_t(t)}. \tag{26}$$

The modulation variable $\boldsymbol{\gamma}(t) \in \mathbb{R}^m$ can be chosen to implement the desired value of $\boldsymbol{y}$. With the above equations, one can easily show that $\dot{S} = u_t^T y_t = \boldsymbol{u}^T \boldsymbol{y}$, and therefore, that power is either injected into or extracted from the tank. As visible in (26), there is a singularity when the tank is empty, and the desired behavior can no longer be implemented. Therefore, it is necessary to ensure that

$$S(t) \geq \epsilon > 0 \quad \forall t. \tag{27}$$

IV. Control Method

In this section, we combine sampling, barrier functions, and energy tank into a novel model-based control framework for robust and safe interaction control.

We propose a *cascaded control* architecture composed of an outer loop where the sampled policy is updated at a low rate and an inner loop where the low-level controller tracks the policy commands. In the outer loop, a sampling-based controller (*reference generation*) generates joints velocity commands over a finite horizon. A sequence of QPs, named *Sequential FILTER-QP*, is solved to *filter* the commands and ensure that motion constraints are fulfilled over the horizon using CBFs. In the inner loop, a similar QP, simply named *FILTER-QP*, is solved to find the best policy command that also satisfies safety and stability constraints using barrier functions and a virtual energy tank.

1) The *Sequential FILTER-QP* takes as input the command sequence $\tilde{U}_t^*$ generated by the sampling-based controller (*reference generation*) and the starting state $\boldsymbol{x}_t$ and returns a modified command-state sequence satisfying all constraints $\bar{U}_t^*$. In turn, the new reference trajectory $\bar{U}_t^*$ is used to warm start the sampling procedure, thus informing the controller with a safe initial policy guess.
2) The *FILTER-QP* takes, at each time step, a single input command, linearly interpolating the reference command trajectory coming from the outer loop. It enforces motion and passivity constraints solving a single QP given the

[3]As shown in [20], CBFs induce an asymptotically stable behavior toward the safe set $\mathcal{C}$. This property is particularly useful when disturbances bring the system outside the constraints.

[4]In comparison, Gurriet et al. [23] address this issue by finding *viable sets*. A set is said to be *viable* if there exists a feasible input, which makes the set forward invariant at all times. Efficient computation of viable sets is still an open research area and is mostly applied to simpler systems due to the high computational requirements.

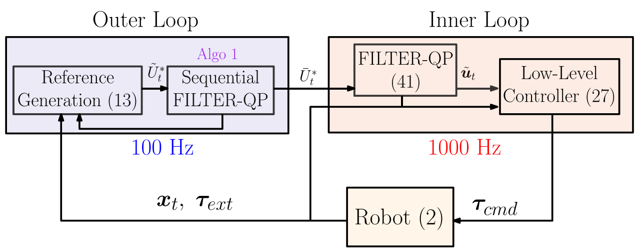

Outer Loop
Inner Loop
Algo 1
Reference Generation (13)
Sequential FILTER-QP
FILTER-QP (41)
Low-Level Controller (27)
100 Hz
1000 Hz
Robot (2)
$\tilde{U}_t^*$
$\tilde{\boldsymbol{u}}_t$
$\boldsymbol{x}_t, \boldsymbol{\tau}_{ext}$
$\boldsymbol{\tau}_{cmd}$

Fig. 2. High-level schematic of the method. The outer loop generates safe trajectories at a low rate, while the inner loop tracks this reference enforcing that passivity and motion constraints are enforced at a higher rate. The outer loop block, including the details of the sampling procedure, is further expanded in Fig. 5.

latest received state and output a single filtered velocity reference $\tilde{\boldsymbol{u}}_t$ for the low-level controller. Since we solve the QP for a single time step, the problem is not computationally expensive and can be solved at a high rate.

Finally, the reference joint trajectory is tracked by a low-level controller that sends raw torque commands to the robot. The complete high-level schematic of the full framework is depicted in Fig. 2.

In the following, we present each component of the framework. We start by presenting our cost formulation for the manipulation task. This is used in the reference generation sampling procedure to weight different trajectory samples and drive the robot to a successful task execution.

Afterward, we encode safety requirements in the form of ZBFs. The safety objectives introduced here are formulated on a kinematic level (obstacle avoidance, joint, and Cartesian limits).

We are also interested in the robustness of the dynamical system during interaction, especially in cases of unexpected events that could compromise its stability. To this end, we complete the proposed framework with a passivity analysis and use an energy tank to bound the energy dissipated during the manipulation task. Passivity is guaranteed in the form of an additional constraint in the quadratic program and, therefore, naturally fits the proposed framework.

We conclude with the description of the deployed low-level controller, which tracks joint velocity references.

A. Reference Generation

The sampling-based controller samples reference commands for the low-level reference tracking controller. We decide to directly sample in the joint velocity space to account for objectives, which are not in the operational control space such as joint limits and self-collision avoidance. The internal model used to sample trajectory rollouts has the same form as in (2).

As described in Section II, the control objective is to drive the system to minimize a task-dependent cost function over time. Furthermore, as state constraints are not explicitly taken into account by the formulation, a common heuristic is to penalize deviations from feasible states in the cost function. In the following, we define several cost components associated with the different high-level objectives and constraints. We denote by $\mathbb{1}[\boldsymbol{x}]$ the *indicator function* such that

$$\mathbb{1}[\boldsymbol{x}]_i = \begin{cases} 1, & \text{if } x_i \text{ is true} \\ 0, & \text{otherwise} \end{cases} \tag{28}$$

where $\star_i$ is the ith element of vector $\star \in \mathbb{R}^n$. In the following, we drop from the notation the dependence on the current state. We use $\boldsymbol{W}$ to denote positive-semidefinite weight matrices and w for nonnegative scalar parameters. Note that input constraints are not treated here as additional cost terms as the nonlinear dynamics can be augmented with a function that projects the sampled inputs into the feasible set $\mathcal{U}$.

1) Target Reaching: In the target reaching task, the goal is to bring a frame attached to the robot (generally the end-effector frame) to a desired pose. We define with $\boldsymbol{T} \in SE(3)$ and $\boldsymbol{T}^* \in SE(3)$ the current and desired target frame pose, respectively. The *tracking cost* is computed as a weighted distance in the tangent space to $SE(3)$ using the logarithmic mapping [36]

$$c_t = ||\log\left(\boldsymbol{T} - \boldsymbol{T}^*\right)||^2_{\boldsymbol{W}_t}. \tag{29}$$

2) Collision Avoidance: Let $\gamma \in \{0, 1\}$ represent an auxiliary variable, which is equal to 1 when the manipulator is in contact with the environment. The value of γ can be computed searching for collisions between bodies. This cost term is activated when we want to execute a contact-free motion of the end-effector. The *contact cost* is defined as

$$c_\gamma = w_\gamma \gamma(\boldsymbol{x}). \tag{30}$$

3) Joint Position Limits: The manipulator is subject to physical joint limits. These can be addressed by introducing a cost component that penalizes violation of the constraints. We define the *joint limits cost* with

$$\begin{aligned} c_j &= \mathbb{1}[\boldsymbol{q} > \boldsymbol{q}_{\text{upper}}]^T \left(w_j + ||\boldsymbol{q}_{\text{upper}} - \boldsymbol{q}||^2_{\boldsymbol{W}_{js}}\right) \\ &\quad + \mathbb{1}[\boldsymbol{q} < \boldsymbol{q}_{\text{lower}}]^T \left(w_j + ||\boldsymbol{q} - \boldsymbol{q}_{\text{lower}}||^2_{\boldsymbol{W}_{js}}\right) \end{aligned} \tag{31}$$

where the scalar w_j is a constant cost added when the limit is violated. The matrix $\boldsymbol{W}_{js}$ adds a quadratic term in the limit violation. This was shown in [37] to help the controller find its way back if poor sampling brings the system outside of the joint position limits.

4) Arm Reach: We introduce an additional term that penalizes configurations where the arm's end-effector moves excessively relative to the base 2-D placement. This helps to bias solutions where base motion is preferred over stretching the arm, which can lead to singular configurations. The translation vector from the end-effector frame $\mathcal{E}$ to the arm base frame $\mathcal{B}$ is defined with the vector $\boldsymbol{p}_{\mathcal{BE}} \in \mathbb{R}^3$. The current reach is then $r_{\text{curr}} = \boldsymbol{p}_{\mathcal{BE}}^T \boldsymbol{P} \boldsymbol{p}_{\mathcal{BE}}$. The projection matrix $\boldsymbol{P} = \text{diag}(1, 1, 0) \in \mathbb{R}^{3\times 3}$ makes sure that the reach is only computed in the 2-D plane. Given a maximum reach $r_{\max} \in \mathbb{R}_{\geq 0}$, the *reach cost* is then defined by

$$c_r = \mathbb{1}\left[r_{\text{curr}} > r_{\max}\right] \left(w_r + w_{rs}\left(r_{\text{curr}} - r_{\max}\right)^2\right). \tag{32}$$

5) Self-Collision Avoidance: Similarly to arm reach, self-collision avoidance can be implemented as an additional cost term, which is active when the distance between a pair of frames is less than a pair-dependent threshold. Given the distance between two frames d_{ij} and a threshold $d_{ij}^{\min}$ the self-collision cost for the pair is

$$c_{\text{sc}} = \sum_{ij, i \neq j} \mathbb{1}\left[d_{ij} < d_{ij}^{\min}\right] \left(w_r + w_{rs}\left(d_{ij}^{\min} - d_{ij}\right)^2\right). \quad (33)$$

6) Object Manipulation: In the manipulation task, the goal is to change the state of an articulated object through interaction. The *manipulation cost* penalizes deviations from the target object configuration $\boldsymbol{o}^*$

$$c_o(\boldsymbol{o}; \boldsymbol{W}_o) = ||\boldsymbol{o} - \boldsymbol{o}^*||^2_{\boldsymbol{W}_o}. \quad (34)$$

7) Power Minimization: We propose a new cost component that takes into account the power dissipated to perform the task. In fact, a successful interaction (i.e., opening the door) could happen in multiple ways; however, trajectories that dissipate low power are most efficient as they do not act against the environment and robot kinematic constraints. We leverage the fact that as rollouts are performed in simulation, the joint torque generated through interaction can be easily computed by summing the contribution of each force $\boldsymbol{f}_c \in \mathbb{R}^3$ at each contact point c

$$\boldsymbol{\tau}_{\text{ext}} = \sum_c \boldsymbol{J}_c^T \boldsymbol{f}_c \quad (35)$$

where $\boldsymbol{J}_c$ is the contact Jacobian. The power dissipated during the task is, therefore, $-\boldsymbol{\tau}_{\text{ext}}^T \boldsymbol{u}$, which is positive when acting "against" the environment. The cost associated with power penalization is

$$c_p = w_p \cdot \max\left(0, -\boldsymbol{\tau}_{\text{ext}}^T \boldsymbol{u} - p_{\max}\right) \quad (36)$$

where $p_{\max}$ is the maximum power that can be dissipated during the task.

8) Cost Scheduling: The manipulation task consists of two phases. In the first phase, the manipulator reaches an estimated contact point, allowing for a fast and successful exploration in the following interaction phase. In the second phase, the goal is to bring the object to the desired state while keeping the end-effector close to the contact location. This switch is manually enabled by turning on the object manipulation cost c_o after a successful approach. Reducing the end-effector position penalty during the manipulation phase allows the controller to choose a trajectory that fully exploits the contact dynamics by changing the hand pose.

B. Safety Constraints

In the previous subsection, a combination of cost components was introduced in order to address both performance and safety. The variety of objectives makes the cost landscape highly complex such that trading off performance against safety objectives can be quite challenging and tedious. Furthermore, as already stressed, sampling does not provide any formal guarantee that constraints encoded in the form of additional cost terms will be satisfied. Therefore, we look at how barrier functions can encode safety-critical constraints. As described in [24], a simple ZBF can be derived for each joint to keep it between its lower and upper bounds, q_i^- and q_i^+, respectively:

$$h_{ql}^i = \frac{\left(q_i^+ - q\right)\left(q - q_i^-\right)}{\left(q_i^+ - q_i^-\right)}. \quad (37)$$

In the following, we treat the safety requirements associated with robot frames and denote with $\boldsymbol{p}_{\mathcal{A}} \in \mathbb{R}^3$ the position of a generic robot frame $\mathcal{A}$ computed through forward kinematics. The ZBF constraint can then be derived using the differential kinematics equation

$$\dot{\boldsymbol{p}}_{\mathcal{A}} = \boldsymbol{J}_{\mathcal{A}}^{\text{lin}} \dot{\boldsymbol{q}} \quad (38)$$

with $\boldsymbol{J}_{\mathcal{A}}^{\text{lin}}$ being the linear Jacobian associated with frame $\mathcal{A}$ and assuming that joint velocities are accurately tracked.

The self-collision safe set can be obtained by approximating potentially colliding frames with nonintersecting spheres. Then, the self-collision ZBF is defined as

$$h_{sc}^{ij} = \frac{1}{2}\left(||\boldsymbol{p}_{\mathcal{C}_i} - \boldsymbol{p}_{\mathcal{C}_j}||^2 - d_c^2\right) \quad (39)$$

where $d_c = r_i + r_j$ is the sum of the radius of the two collision spheres associated with the ith and jth collision pair frames. Note that we can similarly encode arm reach limits. In fact, the following is a valid ZBF, positive only when the end-effector is within the prescribed maximum reach $r_{\max}$ with respect to the arm base

$$h_{\text{ar}} = \frac{1}{2}\left(r_{\max}^2 - (\boldsymbol{p}_{\mathcal{E}} - \boldsymbol{p}_{\mathcal{B}})^T \boldsymbol{P}(\boldsymbol{p}_{\mathcal{E}} - \boldsymbol{p}_{\mathcal{B}})\right). \quad (40)$$

The projection matrix $\boldsymbol{P} = \text{diag}(1, 1, 0) \in \mathbb{R}^{3\times 3}$ makes sure that the reach is only computed in the 2-D plane. This prevents the arm from stretching out and reaching singular configurations. Each of these ZBFs translates to a constraint of the form in (21), which is affine in the commands.

As described in Section III, we can formulate a quadratic program to find the command that is the closest to the sampled one while also satisfying the ZBF constraints previously introduced

$$\begin{aligned} \min_{\tilde{\boldsymbol{u}}_t, \boldsymbol{\delta}} \quad & ||\tilde{\boldsymbol{u}}_t - \boldsymbol{u}_t||^2 + \boldsymbol{\delta}^T \boldsymbol{\Gamma} \boldsymbol{\delta} \quad \text{(FILTER-QP)} \\ \text{s.t.} \quad & \dot{h}_{ql}^i \geq -\gamma h_{ql} + \delta_{ql}^i \quad \forall\, i \in [1, m] \text{ (Joint Limits)} \\ & \dot{h}_{sc}^{ij} \geq -\gamma h_{ij} + \delta_{sc}^{ij} \quad \forall\, (i,j) \in \mathcal{I} \text{ (Self Collision)} \\ & \dot{h}_{ar}^i \geq -\gamma h_{ar} + \delta_{ar}^i \text{ (Arm Reach)} \\ & \tilde{\boldsymbol{u}}_t \in \mathcal{U} \text{ (Input Limits)} \end{aligned} \quad (41)$$

where $\boldsymbol{\delta}$ is the vector of slack variables and $\boldsymbol{\Gamma}$ is a positive-definite diagonal matrix weighting the slack variable penalization whose dimensions depend on the number of implemented soft constraints.

Remark 1: Note that slack variables might cause a violation of the constraint. We address this problem using more conservative limits, allowing for a small constraints violation at the constraint boundary. In particular, if $\tilde{h}$ is the original ZBF, we deploy a stricter barrier function $h = \tilde{h} - \Delta$ with $\Delta \geq 0$. Imposing that the slack variables are bounded from below

$$\delta > -\gamma\Delta \quad (42)$$

Algorithm 1: Sequential FILTER-QP.

```
Data: optimal input sequence $U_t^* = \{\boldsymbol{u}_t^*, \dots, \boldsymbol{u}_{t+T-\Delta t}^*\} = \{\boldsymbol{\mu}_t, \dots, \boldsymbol{\mu}_{t+T-\Delta t}\}$ ,
      current state $\boldsymbol{x}_t$, simulation steps $K$
Result: filtered input trajectory $\bar{U}_t^*$
$\boldsymbol{x} \leftarrow \boldsymbol{x}_t$;
$\boldsymbol{u} \leftarrow \boldsymbol{u}_t^*$;
$k \leftarrow 0$;
while $k < K$ do
    $\bar{\boldsymbol{u}}_{t+k\Delta t} \leftarrow$ FILTER-QP$(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\tau}_{ext})$    (Eq. 41)
    $\boldsymbol{x}, \boldsymbol{\tau}_{ext} \leftarrow$Step Simulation    (Eq. 2)
    $k = k + 1$
    $\boldsymbol{u} \leftarrow \boldsymbol{u}_{t+k\Delta t}^*$
```

the ZBF constraints can be rewritten as

$$\dot{\tilde{h}} = \dot{h} \geq -\gamma h + \delta = -\gamma \tilde{h} + \gamma \Delta + \delta \geq -\gamma \tilde{h}. \tag{43}$$

Therefore, satisfying the constraint on a stricter set *with* bounded slack variables implies satisfying the hard ZBF constraint on the original larger invariant set

$$\dot{h} \geq -\gamma h + \delta \quad \Rightarrow \quad \dot{\tilde{h}} \geq -\gamma \tilde{h}. \tag{44}$$

One can tune the stricter safe set such that the slack variable does not get smaller than the safety margin $-\gamma\Delta$. Practically, for our conservative choice of safe set, the constraints on the slack variables are always satisfied; thus, we can solve the simpler optimization in (41).

Finally, the set denoted by $\mathcal{I}$ is the set of collision link pairs. We name this problem FILTER-QP as it changes the control input only if it is unsafe. Instead of simply applying this optimization pointwise, we can *filter* the full input sequence in a sequential manner. After a policy update and for each time step, a FILTER-QP is solved. The newly computed command is applied to advance to the next step in the horizon and obtain the next system state from which the constraint equation can be updated. We call this procedure Sequential FILTER-QP and describe it in Algorithm 1. The resulting filtered optimal input sequence $\bar{U}_t^*$ is then also used to warm-start the nominal policy for the next round of rollout sampling (see Fig. 5). The reader can also refer to Fig. 2 for a schematic representation of the complete feedback control loop.

C. System Stability

While the optimization in (41) enhances the safety of the system, we still lack stability guarantees. As described in Section III, energy tanks can be used to make the system *passive* and, thus, ensure stability. Inspired by the work in [24] and [30], this adaptation naturally fits the control method we have developed so far. If we augment the optimization problem in (41) with the energy constraint

$$\int_0^t \boldsymbol{\tau}_{\text{ext}}^T \dot{\boldsymbol{q}}^* \, d\tau \geq -S_{\text{tank}}(0) \tag{45}$$

where $S(0)$ is the initial energy stored in the tank, then the following stability result holds.

Theorem 1 (Controlled system stability): If the optimization problem FILTER-QP satisfies constraint (45), then the controller makes the system (2) passive and, therefore, stable.

Proof: We start by augmenting the system model with a virtual tank. We now need to define through which *power ports* the tank exchanges energy with the rest of the system. To this end, we perform a passivity analysis of the *real* system. We first define the robot energy as

$$S_{\text{robot}} = \frac{1}{2}\dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{M}(\boldsymbol{q})\dot{\tilde{\boldsymbol{q}}} + \frac{1}{2}\tilde{\boldsymbol{q}}^T \boldsymbol{K}_I \tilde{\boldsymbol{q}}. \tag{46}$$

In order to study the passivity of the system, we need to derive the robot energy dynamics $\dot{S}_{\text{robot}}$. A complete derivation can be found in Appendix B. It turns out that

$$\dot{S}_{\text{robot}} = \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{\tau}_{\text{ext}} - \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{K}_D \dot{\tilde{\boldsymbol{q}}} \leq \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{\tau}_{\text{ext}}. \tag{47}$$

The power flow through the external torque has indefinite sign and can lead to a loss of passivity. Since the environment is passive, there exists an environment energy S_{env} such that [30]

$$\dot{S}_{\text{env}} \leq -\dot{\boldsymbol{q}}^T \boldsymbol{\tau}_{\text{ext}}. \tag{48}$$

The energy tank is finally connected to the system through the power port $(\dot{\boldsymbol{q}}^*, \boldsymbol{\tau}_{\text{ext}})$ and therefore

$$\dot{S}_{\text{tank}} = \dot{\boldsymbol{q}}^{*T} \boldsymbol{\tau}_{\text{ext}}. \tag{49}$$

Then, the energy evolution of the autonomous system, composed of robot, tank, and environment, is

$$\begin{aligned} \dot{S}_{\text{tot}} &= \dot{S}_{\text{robot}} + \dot{S}_{\text{tank}} + \dot{S}_{\text{env}} \\ &\leq \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{\tau}_{\text{ext}} + \dot{\boldsymbol{q}}^{*T} \boldsymbol{\tau}_{\text{ext}} - \dot{\boldsymbol{q}}^T \boldsymbol{\tau}_{\text{ext}} \leq 0 \end{aligned} \tag{50}$$

showing the system's passivity. Intuitively, the increase of the robot's energy is compensated with a reduction of the tank's energy. Therefore, the total energy stored in the system is bound by $S_{\text{tank}}(0) + \Lambda$ with Λ some positive constant. In order for (50) to be valid, we are relying on the silent assumption that the robot is able to exchange energy with the tank at any time. As the tank energy is a positive-definite quantity and therefore bounded from below, we need to ensure that it is never completely depleted, imposing

$$S(t) = S(0) + \int_0^t \boldsymbol{\tau}_{\text{ext}}^T \dot{\boldsymbol{q}}^* \, d\tau \geq 0 \tag{51}$$

recovering the constraint in (45), therefore concluding the proof. In other words, (51) is a condition that is required for (50) to hold at any time. □

In practice, the constraint in (45) is formulated as

$$\int_0^t \boldsymbol{\tau}_{\text{ext}}^T \boldsymbol{u} \, d\tau \geq \epsilon - S(0) \tag{52}$$

where ϵ is a positive minimum residual energy to avoid the singularity that the tank suffers when it is completely depleted.

D. Low-Level Controller

Finally, we describe the low-level control law. We consider a dynamic manipulator as described in (2). The robot uses a dynamically compensated low-level velocity controller to convert a velocity reference in torque commands

$$\boldsymbol{\tau}_{\text{cmd}} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}^* + \boldsymbol{C}_r(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}^* + \boldsymbol{g}_r(\boldsymbol{q}) - \boldsymbol{K}_D\dot{\tilde{\boldsymbol{q}}} - \boldsymbol{K}_I\int_0^t \dot{\tilde{\boldsymbol{q}}}\, d\tau \tag{53}$$

with the auxiliary error variable $\tilde{\boldsymbol{q}} = \boldsymbol{q} - \boldsymbol{q}^*$ and $\boldsymbol{K}_D$ and $\boldsymbol{K}_I$ being positive diagonal gain matrices.

V. Practical Aspects

We address here some issues that arise when implementing the described algorithm on a resource-constrained platform. Furthermore, we introduce many practical expedients that deal with limited sampling budget, slow control rates, and finally with sim-to-real transfer.

A. Gradient Clipping

The policy update rule in (14) consists of a receding horizon *mini-batch* SGD. As a consequence, the gradient variance can vary greatly between successive iterations of the algorithm. To prevent this phenomenon, which is especially evident with a small sampling budget, we clip the gradients to a user-defined threshold

$$g_i = \text{sign}(g_i)\cdot\min(|g_i|, g_{\max}) \tag{54}$$

where g_i is the ith component of the stochastic approximation of the gradient vector and $g_{\max} \in \mathbb{R}_{>0}$ is the gradient threshold.

B. Likelihood Mapping

The coefficient λ in (15) determines how "aggressive" the weighting between different trajectories is. Adopting a constant value would give a numerically zero weight to most of the trajectories. Shifting the trajectory cost by the minimum cost as proposed in [15] also does not alleviate this issue. Especially, in regions of high cost, trajectories that are locally near optimal can be assigned very different weights. We instead propose to adopt the same technique as in [38], where λ is scaled to better discriminate between the experienced trajectories. The modified exponential utility is then defined by

$$\exp(-\lambda J) = \exp\left(-h\frac{J - J_{\min}}{J_{\max} - J_{\min}}\right) \tag{55}$$

which has the nice property of being invariant to the cost scale. Assume, for example, that in a target reaching task, the goal pose is far away, and thus, the cost incurred by each sampled rollout is scaled by a common scalar factor. This factor would disappear when using the previous equation for mapping rollouts to likelihood. We show the effect of cost-scale invariance in Fig. 3. In the figure, we assume that sample costs are uniformly distributed in the range between minimum and maximum cost and that the minimum cost shows a 20% reduction with respect to the worst rollout.[5] We compare the exponential mapping $\mathcal{J} = \exp(-\lambda J)$ (*naive*), with baseline reduction $\mathcal{J} = \exp(-\lambda(J - J_{\min}))$ (*baseline*) and the mapping in (55) (*invariant*).

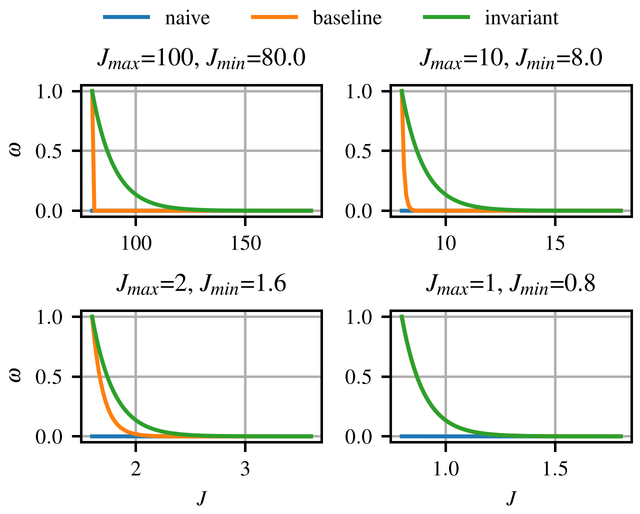


Fig. 3. In each plot, we change the maximum and minimum cost (80% of maximum cost). We set λ and h to 10 in all comparisons.

When the cost is high, the first two methods collapse all the weight to a single sample, and in practice, other samples are assigned a value that is numerically zero. As a consequence, the gradient estimate in (14) can show a high variance. The mapping in (55) instead assigns a nonzero weight to more samples independently from the current cost scale and helps to stabilize the gradient estimate.

C. Chattering

We observed that a naive implementation of the passivity constraint leads to a chattering behavior at the constraint boundary. We start discretizing the constraint inequality to obtain a constraint that is affine in the input $\boldsymbol{u}$

$$\int_0^{t+dt} \boldsymbol{\tau}_{\text{ext}}^T\boldsymbol{u}\, d\tau \approx \underbrace{\boldsymbol{\tau}_{\text{ext}}(t)^T\boldsymbol{u}(t)}_{-P_{\text{diss}}} dt + S(x_t(t)) \geq \epsilon$$

which is equivalent to

$$P_{\text{diss}} \leq \alpha E_{\text{res}} \tag{56}$$

where we defined the residual energy $E_{\text{res}} = S(x_t(t)) - \epsilon$. The above inequality has the drawback that when the Euler integration is performed with small time intervals, then a dangerously high power can be dissipated at any time, until very close to the lower energy limit, thus leading to chattering. It is, therefore, better to use a value of $\alpha < 1/dt$. Furthermore, one can easily verify that defining a passivity ZBF as $h_{\text{pass}} = S(x_t) - \epsilon$ we obtain the same formulation. The ZBF constraint reads as

$$\dot{h}_{\text{pass}} = \dot{S}(x_t) \geq -\alpha h_{\text{pass}} = \alpha(\epsilon - S(x_t)).$$

The passivity can, therefore, be seen as a ZBF and, consequently, inherits its properties. A smaller value for α makes the constraint

[5]This is a reasonable choice since we are in an online setup and we assume to be in a low sampling budget regime.

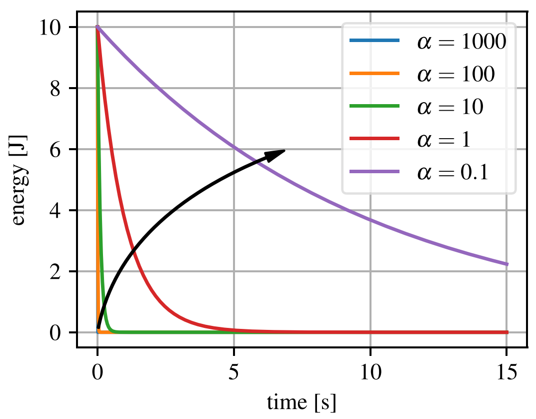


Fig. 4. *Worst case* tank energy profile which is how the energy in the tank would evolve if the maximum allowed energy would be dissipated. The smaller the α, the more conservative the constraint, and thus, the energy is depleted at a lower rate.

more conservative and improves its performance at the boundary.

In order to gain a better intuitive understanding of this tuning parameter, consider the worst case scenario. In this scenario, the power flow is always equal to the maximal dissipated power allowed by the passivity constraint: $\dot{E}_{\text{res}} = -\alpha E_{\text{res}}$. Then, $E_{\text{res}}(t) = E_{\text{res}}(0)e^{-\alpha t}$. A smaller α reduces the energy tank depletion rate (as shown in Fig. 4) at the cost of making the system more conservative. This effect is also shown in Fig. 12 in the experiment section.

D. Sampling Strategy

At each control step, it is desirable to sample a zero input trajectory, meaning that all velocity commands are $\boldsymbol{u}_t = \mathbf{0}\ \forall\, t$, as this would allow the robot to stop when the goal is reached. Similarly, we would like to keep sampling the unperturbed nominal command trajectory. In fact, if we assume this to be optimal for a time instant, resampling would perturb it and very likely increase its associated cost. With these motivations in mind, we modify the sampling procedure such that the unperturbed and zero velocity input sequences are always in the batch of samples.

Furthermore, at the beginning of a new sampling, a subset of the best rollouts is kept and shifted back in time by the real-time number of steps elapsed since the last optimization to warm start the next iteration. As the sampled control is not sufficiently smooth for a practical application, we filter it using a moving window Savitzky–Golay filter [39] before the Sequential FILTER-QP. In fact, the Savitzky–Golay filter efficiently computes the optimal local polynomial approximation of the input sequence, without the need to explicitly solve an optimization problem, as has been shown in [37]. The full pipeline is visualized in Fig. 5.

E. Contact-Mesh Simplification

A large proportion of simulation time is spent on collision detection. We reduce the component meshes to the relevant ones and simplify to primitive shapes, as shown in Fig. 6. We are especially interested in the collision objects belonging to the

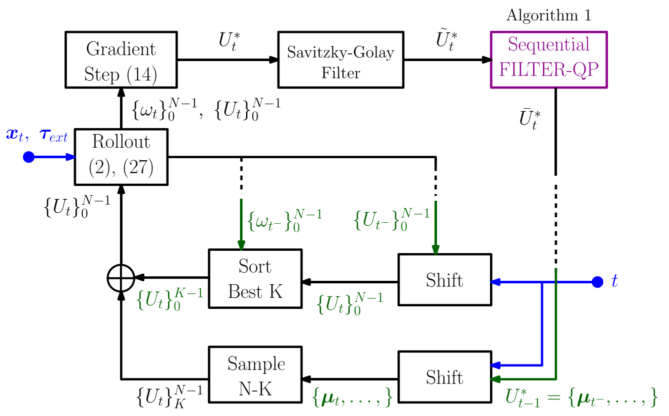


Fig. 5. Implementation of the receding horizon scheme (*reference generation*; see Fig. 2). Variables depicted in *blue* and *dark green* represent the information available at the new optimization step and from the previous iteration, respectively. All the previous samples and the nominal trajectory are shifted by $t - t^-$. The K best input sequences (according to their weights) are reused and concatenated to a batch of $N - K$ fresh samples to perform new rollouts. The Sequential-QP block, highlighted in *purple*, is optional and deployed by different variants of the method.

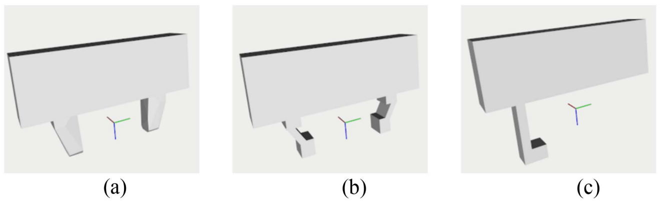


Fig. 6. Original collision meshes (a) provided by the robot manufacturer are often approximated by convex hulls, which are inaccurate while also more complex representations. The mesh in (b) is an exact collision shape for the original gripper made of simple box primitives. We reduce the end-effector complexity by designing a simple single-finger gripper (c) made of only two boxes obtaining a computational performance gain in terms of simulation rate. (a) Original mesh. (b) Two fingers. (c) Hook finger.

robot end-effector and the object. This adaptation tremendously reduces computation especially during the contact phase. The original finger design and vendor mesh is shown in Fig. 6(a). Often, as well as in this case, the vendor mesh is a convex hull of the actual finger geometry. In our evaluation, we have replaced the convex hull with the real mesh, building it with a set of nine box primitives, as shown in Fig. 6(b). Finally, to further reduce simulation time, we redesigned the hand with a single finger hook setup for nonprehensile manipulation, further reducing the needed box primitives to three. We observed that using the original mesh shown in Fig. 6(a) results in a mean simulation rate of 0.98 kHz. In contrast, the two finger mesh and hook finger mesh in Fig. 6(b)and6(c) allow mean simulation rates of 2.13 and 2.94 kHz, respectively.

F. Simulator Tuning

1) Contact-Mesh Simplification: Crucial to the overall performance is the accuracy of the simulation environment [implementing (2)]. Unfortunately, the discrepancy between the simulator and the real physical model known as the *sim-to-real* gap is unavoidable. A typical failure case consists of an overestimation of an object's friction. In such cases, we have often observed a "scratching" emergent behavior where the robot would rely on friction to move the object. In practice, friction is hard to measure and depends on the contact patch between surfaces. On the other hand, kinematic constraints between contact points depend on the system geometry that can be accurately measured (e.g., from computer-aided design models). Therefore, solutions that exploit the latter are more likely to succeed on the real platform. One can bias the controller toward these solutions by setting, for example, a very low friction coefficient between contact bodies.

2) Simulator Stability: A big bottleneck of the proposed approach is the computation required to forward simulate multiple samples of the system dynamics in the physics engine. The time complexity is linear in the number of rollouts, the time steps, and horizon. In order to keep a sufficiently long horizon (1 s) and a good amount of samples, we increase the time discretization for forward integrating the simulated dynamics. However, simulators suffer instability issues when the time steps are large [40]. A direct implementation of (53) in the physics engine would often cause instability during the 1-s time horizon. Fortunately, the efficient simulator provides an internal proportional-derivative controller [41] that allows it to track joint position and velocities with big time steps (0.015 s in this article) without suffering instability. The control law is then a simple P velocity controller with the compensation of Coriolis and gravity terms

$$\boldsymbol{\tau}_{\text{cmd}} = \boldsymbol{C}_r\left(\boldsymbol{q}, \dot{\boldsymbol{q}}\right)\dot{\boldsymbol{q}} + \boldsymbol{g}_r(\boldsymbol{q}) - \boldsymbol{K}_D\dot{\tilde{\boldsymbol{q}}} \tag{57}$$

with an appropriate choice of the positive-definite diagonal gain matrix $\boldsymbol{K}_D \in \mathbb{R}^{n_q \times n_q}$.

Using a different low-level controller to simulate interaction trajectories inevitably introduces a model mismatch. We argue that if the real and simulated robot can track the joint velocities well, namely they have similar closed loop velocity dynamics, then we can abstract away the low-level control layer and look at the two as robots that can be directly commanded in joint velocity.

Furthermore, even when the two low-level control laws would be identical, a mismatch is still potentially possible because of numerical and model errors [40]. Addressing these errors requires a comparative study of many simulators and involves system identification, generally a harder problem than simulation, and is outside the scope of our work. Qualitatively, our approach is a good compromise between theoretical guarantees (deploying (53) on the real robot), feasibility, and simulation stability [deploying (57)] on the simulated robot.

VI. Numerical and Experimental Evaluation

The full method is amenable to different variants. In particular, by turning ON/OFF filtering in the outer or inner control loop, we can synthesize four possible controller implementations. In the following, we define with Π_* each controller, where $*$ can be N, I, O, IO

1) Π_N: It relies exclusively on a stochastic controller to generate velocity commands, namely, the Sequential FILTER-QP and FILTER-QP are completely missing and the sampled input trajectory is tracked as it is by the low-level controller.

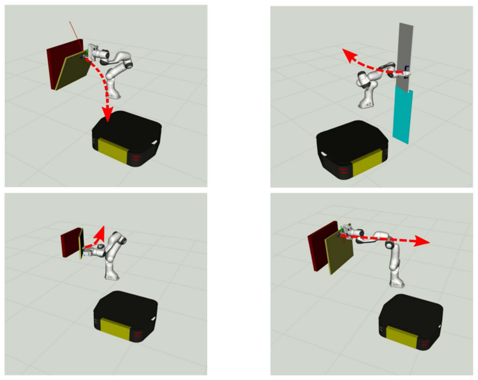


Fig. 7. Four articulated objects used in our simulation evaluations. From left to right: *shelf*, *dishwasher*, *microwave*, and *drawer*. A red arrow shows the expected end-effector trajectory to perform the task.

2) Π_I: a FILTER-QP is deployed in the inner control loop to filter the instantaneous input command to be tracked by the low-level velocity controller.
3) Π_O: the Sequential FILTER-QP is used in the outer loop sampling controller to filter the full optimized command trajectory in a sequential manner.
4) Π_{IO}: The full cascaded architecture is deployed, combining the previous two methods.

The goal of this section is to evaluate the proposed control methods and all the various implementation previously introduced, highlighting the contribution of each algorithmic component. To this end, we define the following metrics.

1) *Cumulative constraint violation:* As each barrier function is, by definition, negative outside of the safe set, we define the following metric as a proxy for the size of the constraint violation along the duration of an experiment:

$$\Delta^i_{\text{tot}} = \sum_{k=0}^{K_{\text{exp}}} \max\left(0, -h^i\left(\boldsymbol{x}_{k\Delta t}\right)\right)$$

referring to the ith ZBF, where $K_{\text{exp}} = \lfloor T_{\text{exp}}/\Delta t \rfloor$.
2) *Average interaction wrench:* Average wrench that is exerted on the environment during the execution of the task.
3) *Dissipated power:* During an ideal interaction with an articulated object, power is minimally dissipated. Therefore, we use the dissipated power as an efficiency metric

$$P_{\text{diss}} = \sum_{0}^{T_{\text{exp}}} -\boldsymbol{u}^T\boldsymbol{\tau}_{\text{ext}}. \tag{58}$$

We always report a successful task execution, making unnecessary a success rate metric. The simulation experiments are conducted on a dynamic manipulator model, as described by (2). The manipulation tasks consist of maneuvering different articulated objects: a *shelf*, a *dishwasher*, a *microwave*, and a *drawer*, as shown in Fig. 7. They differ in type and orientation of the joint. The *shelf* and *microwave* have a vertical revolute joint, the *dishwasher* has a horizontal revolute joint, while the *drawer* has a horizontal prismatic joint.

The control methods are deployed on the RoyalPanda mobile manipulator and evaluated through simulated and hardware experiments. The platform consists of a holonomic mobile base equipped with a seven-DOF manipulator. The robot's wrist mounts a six-axis force–torque sensor and a custom set of fingers, as shown in Fig. 6(c). The omnidirectional base is controlled by sending velocity commands to the mecanum wheel controller. The arm's low-level controller runs at 1 kHz, while the base mecanum controller runs at 50 Hz. In both the real and simulated robots, the arm velocity commands are converted to joint torques using a low-level proportional–integral velocity controller running at 1 kHz. An open-source implementation of the proposed algorithms, supporting multithreaded sampling and generic robotic systems, is available at https://git.io/JXi4d.

A. Comparison of Methods

We design the following experimental scenarios.

1) *Object manipulation:* We validate the applicability of the method on the task of manipulating each of the four articulated objects. In each environment, there is a handle that the robot can use to move the articulated object. In a first step, the end-effector reaches the handle and then tries to make contact and move the articulation to the desired position. The switch between hand reaching and object manipulation is triggered by turning on the object reference cost, as described in Section IV.
2) *Target reaching and obstacle avoidance:* We further investigate the validity of the constraints formulation for a target reaching task in a confined environment. The end-effector must reach a target goal, while the base is constrained in a narrow corridor. Furthermore, the robot must avoid an obstacle represented as a red sphere in Fig. 9. The obstacle is not immediately visible but appears all of a sudden, challenging the reactive response of the sampling-based controller.
3) *Robust interaction:* We construct a challenging manipulation scenario where robust interaction is needed; in this case, the articulated object is fixed rigidly and is not able to move for a short period of time.
4) *Real-world experiments:* We further test the methods on the real robot for a door opening task.

The *reference generation* module outputs a discretized 1-s horizon trajectory at ca. 100 Hz. The trajectory is discretized using a discretization time interval of 0.015 s, resulting in ca. 66 steps. The low-level *reference tracking* controller then extracts a joint velocity reference from the model predictive path integral control (MPPI) trajectory which is the closest in time to the current time step at the frequency of 1 kHz. Thus, the input is oversampled by linearly interpolating between subsequent time steps. The joint reference values are then filtered using the FILTER-QP and then converted to joint torque commands using (53). We run the presented algorithms on an Intel Core i7-8550 U quad-core processor (1.8 GHz, up to 4.0 GHz) and use eight threads for parallel forward sampling of rollouts. Forward simulation is provided by the `raisim` physics engine [42]. The FILTER-QP and Sequential FILTER-QP are solved efficiently

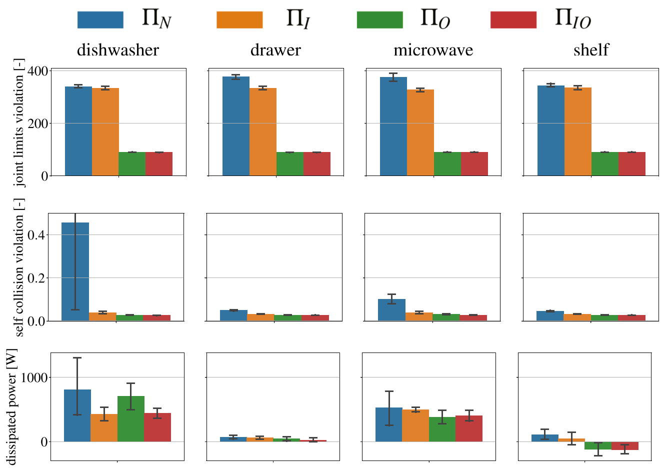


Fig. 8. Comparison between the different control methods. Note that the self-collision bar plot has a data point at 3.4 in the Π_N method for the *dishwasher* case, which is far outside the plot range.

using the `osqp` C++ library [43]. We measure an average solving time of ≈ 0.1 ms. A summary of the main control parameters is presented in Table I.

Note that the goal of this section is to evaluate the validity of the framework. Nonetheless, similar experiments could be conducted with a combination of an admittance controller and a reference generator. However, a stochastic controller already samples in joint space, thus optimizing for all DoFs, while also accounting for constraints violation. An admittance controller would instead require an extra redundancy resolution module.

1) Object Manipulation: We would like to investigate the performance of the algorithms when working close or outside the safety boundaries. To this end, in all the simulated experiments, the robot starts in a configuration that is near the arm's joint limits and self-collision. The base of the robot is at $(-3.0, -3.0)$, outside of the prescribed position limits of $[(2.0, 2.0), (-2.0, -2.0)]$. The task of the robot is to open an articulated object moving from its starting location. We perform 20 runs for each articulated object and for each control method.

The results of the experiments are summarized in Fig. 8. We observe a reduction of cumulative joint limit violations as well as self-collision violations. We deduce that filtering the input sequence has a beneficial effect. Π_O and Π_{IO} attain the best performance suggesting that in a low sampling regime, the optimization problem helps to adapt the input sequence to quickly reduce the amount of constraint violation. The second row of Fig. 8 also shows that the naive controller Π_N even experiences a dramatic failure, which is not visible because of plot limits. This edge case never happens for the other methods.

The dissipated power is similar across the proposed methods although filtering seems to also have a positive effect in this aspect. A qualitative analysis has shown that the interaction wrench measured during a simulated rollout can be inaccurate, especially when the rollouts use big time steps (0.015 s) in order to trade off simulation accuracy with speed. As the wrench information is used to evaluate the system passivity, we activate the passivity enforcing constraint only in the outer-loop FILTER-QP since this optimization problem consumes real wrench measurements at high rates. This choice is additionally justified by the fact that passivity is mainly a mechanism designed to react to unforeseen events. As these are not modeled, the simulated rollouts are not able to predict the true evolution of the interaction wrench, and therefore, when passivity is enforced throughout the rollouts, it could lead to detrimental effects, namely an implementation of a nonpassive behavior on the real robot.

Overall, the simulated experiments confirm the validity of the full framework (as implemented in Π_{IO}). Nevertheless, the performance differences are not striking. We think that this is due to the low chance of hitting the constraints during the task, especially when the sampling-based controller is aware of dangerous configurations through a well-engineered cost function. In fact, high cost on safety-related objectives has the effect of "trimming" bad trajectories, removing the need for the postprocessing performed by the constrained optimization problem. For this reason, in the following, we present a second simulation experiment, which represents a challenge for a purely sampling-based controller.

2) Target Reaching and Obstacle Avoidance: The naive stochastic controller relies on a task-encoding cost formulation and sampling to generate "good" rollouts. This method faces two main challenges.

1) The cost needs to be nicely tuned in order to prevent edge cases where performance is chosen in lieu of safety.
2) Sudden changes in the cost function lead to a drastic change in the policy distribution.

While the first issue can be addressed by tuning the cost with a trial-and-error method, the second is more subtle. In fact, the policy should be able to quickly adapt but this can be hard to achieve when only sampling around the previous (outdated) input distribution. In this experiment, we reproduce the described issue during a target reaching task. We place a collision sphere at each robot link and an obstacle in the robot workspace. The base motion is also constrained such that, in order to achieve the goal, the robot is forced to avoid the obstacle while going through a narrow passage. The obstacle is perceived only when the robot is very close to it (< 1 cm) causing a sudden change in the cost function. We show the end-effector optimal trajectory for Π_N and Π_{IO} after the obstacle has been detected in Fig. 9. We can see that the naive controller Π_N is not able to quickly adapt to the unforeseen cost change and instead is trapped in a high-cost region where it is hard to find a good tradeoff between obstacle avoidance and the target reaching objective. In contrast, the Π_{IO} controller immediately adapts the input sequence to comply with the constraints, and sampling can later be performed in a more favorable region of the input space. Note that if the controller would not be aware of constraints in the form of additional cost terms, the generated trajectory would not agree with the solution provided by the Sequential FILTER-QP. In turn, this means that the controller would not exploit the filtered trajectories, as they would be "less" optimal based on a purely task-related cost function.

We repeat the simulation for ten runs for each of the methods. The results are summarized in Fig. 10. As expected, Π_{IO} attains

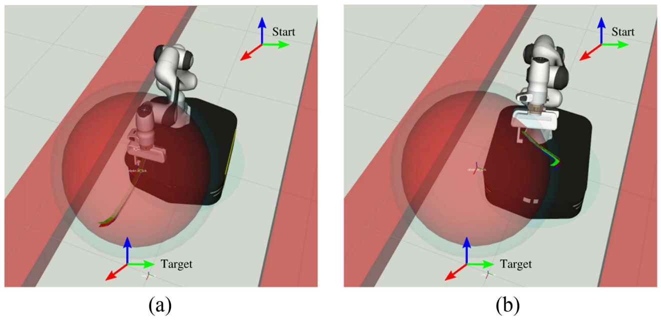


Fig. 9. In this figure, we visualize the optimized end-effector trajectory when the task is to reach a target beyond obstacles represented as transparent-red objects in the simulated scene. Note that the base is also constrained to move through a narrow corridor, visualized by the red markers on each side of the robot. The naive controller Π_N is trapped in a high-cost region, while Π_{IO} immediately reacts to the unexpected cost change. (a) Example rollout of Π_N. (b) Example rollout of Π_{IO}.

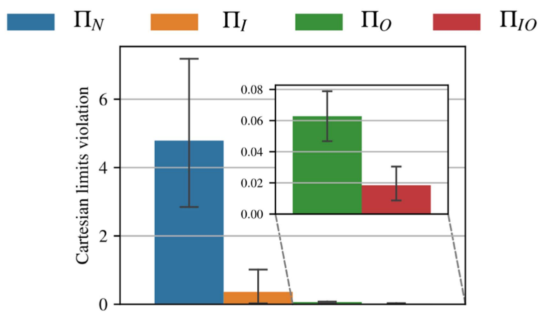


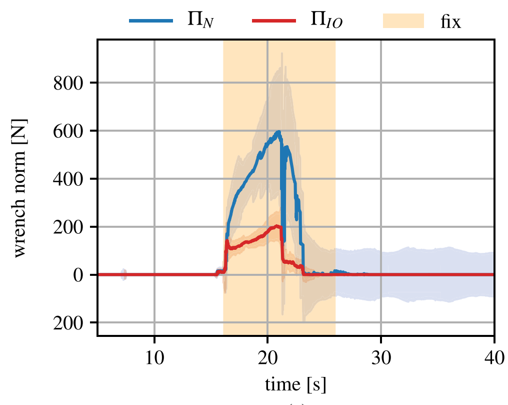
Fig. 10. Cumulative Cartesian limits violation for the target reaching and obstacle avoidance task. $\Pi_O$already outperforms Π_N and Π_I. A further small performance boost is achieved by Π_{IO}.

the best performance. We further observe that the presence of a FILTER-QP in the inner loop additionally improves robustness as it enforces constraints during the open-loop execution of the optimized trajectory received by the stochastic controller.

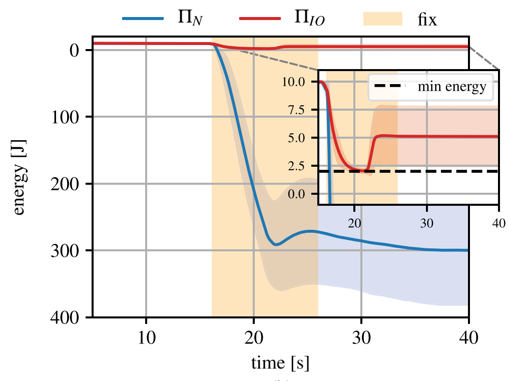
3) Robust Interaction: The previous validations show how the framework can address manipulation tasks and overcome some limitations of a naive stochastic controller. We aim to find evidence that the additional passivity enforcing constraint adds robustness to the method. By adding the energy tank constraint to the optimization problem, we limit the maximum dissipated energy and, thus, generate a stable interaction behavior.

To evaluate this, we alter the *shelf* scenario described earlier such that the motion of the shelf door is limited while the robot is interacting with it. We fix the door position for 5 s, simulating it becoming "stuck." After this time, the object is released and is free to move within its original limits again. This experiment could approximate a real scenario where the object is temporarily constrained because of defects in the opening mechanism or joint wear.

From the results in Fig. 11, we note that when using Π_N, i.e., no passivity is ensured, the negative power flow is not bounded, leading to high interaction wrenches. On the other hand, when the energy tank is deployed, as in Π_{IO}, only a maximal amount of energy, namely that stored in the tank, can be used. Furthermore, we use a small value of $\alpha = 1$ in the



(a)



(b)

Fig. 11. Filter regulates the dissipated power when energy is low in the tank, resulting in a reduced wrench when the object is "stuck." The line plot and shaded area show the mean and standard deviation from 20 simulation runs of interaction with the *shelf* object respectively. (a) *Orange*-shaded area shows the time interval when the object is fixed. Note that this might vary for each experiment as the object reaches the prescribed position at different time points. (b) Energy of the tank when the filter is deployed (*red* curve) is always above the prescribed minimum, while it drastically drops in the other case. The plotted energy is computed integrating the dissipated power during interaction.

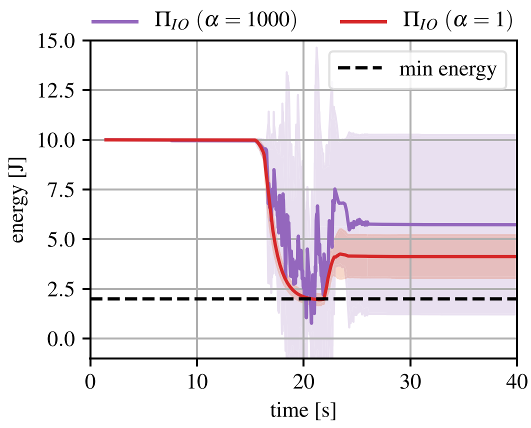


Fig. 12. In this figure, we show the chattering effect happening when the constraints are enforced with $\alpha = 1/dt = 1000$. In contrast, a lower α greatly alleviates this issue.

passivity constraint formulation (56). As discussed in Section V and shown in Fig. 12, this choice allows for a smoother power regulation when approaching the lower energy bound as compared to larger values of α and prevents chattering.

4) Real-World Experiments: The goal of the hardware experiments is to demonstrate that the control framework can be deployed on a real platform in a real-time setup. For this purpose, we perform an interaction replanning and a door opening experiment similar to the description provided in the previous section with our RoyalPanda robot. A video of the experiments can be found https://youtu.be/Q1AohXmsLs4here. The door displacement and the robot's base are tracked using a motion tracking system, eliminating the need for precise state estimation. We plan to remove this limitation in future work.

In the first hardware experiment, the door motion is limited using a rope. With this experiment, we bring interaction to the limit case of extreme force applied to the articulation. Also, this situation might happen when a joint is stuck, e.g., because of stiction, wear, and rust. We monitor the evolution of the tank energy, and once it has reached the allowed minimum, we wait 3 s before manually cutting the rope. We repeat the experiment for the methods Π_N and Π_{IO} recalling that the latter only uses the passivity constraint in the inner loop. The wrench norm and the evolution of the energy in the tank during the experiments can be seen in Fig. 13. As one can see in the accompanying video, without passivity, the manipulator exerts a rapidly increasing wrench until the rope is broken apart. On the other hand, when passivity is enforced, the power flow and external wrench is regulated leading to a robust interaction behavior and no need for the operator to intervene. When the object is released, the gripper gets stuck in a constrained configuration between the door plate and the handle. When Π_N is deployed, the controller tries to push aggressively and is, therefore, not able to escape this gripper trap. Instead, when using Π_{IO}, the tank residual energy is low and aggressive actions are prohibited. The overall behavior is safer and allows the robot to escape from the bad configuration.

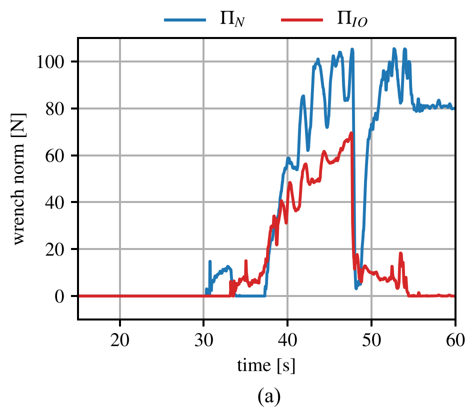
In the second real-robot experiment, contained in the accompanying video, we aim to show the closed-loop and reactive nature of the control framework. During the experiment, the interaction with the articulated object is disturbed by a human operator, forcing the end-effector to lose contact with the object handle. We observe that the controller is able to react and quickly find a new interaction strategy that achieves the task, while staying compliant at all times.

VII. Method Limitations and Future Work

In the following, we describe the most prominent issues that emerged during the experiments and how they could be addressed to improve the proposed method.

A. Model Mismatch

We have found that the method is highly sensitive to model mismatches. While we can assume that the robot model is known with high accuracy, this is often not the case for the object we intend to manipulate. Unfortunately, we observed that a small uncertainty in the estimate of the object pose and geometry could lead to failures in the door opening experiment. In particular, if the handle placement was not measured exactly,



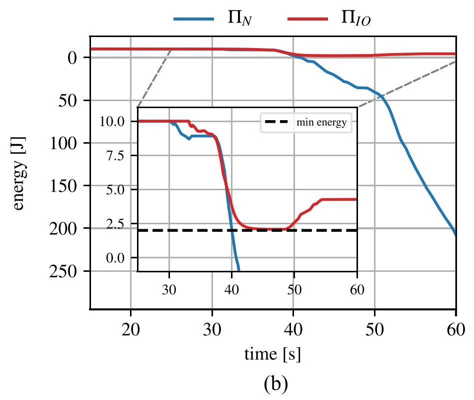


Fig. 13. Figures show that when passivity is enforced, the controller is able to limit the overall interaction wrench. This experiment is also reported in the accompanying video. (a) Wrench norm during the door opening experiment. We measure an average wrench of 68 and 34 N for the Π_N and Π_{IO} methods, respectively. The sharp wrench drop corresponds to the time when the rope breaks and energy is suddenly released. The two time series have been aligned to facilitate visualization. (b) Evolution of the tank energy over time during the door opening experiment. The zoomed plot shows that energy is recovered when the object is released.

the controller would generate a "tapping" motion, which would cause a hard collision between the fingertip and the handle itself. One can try to quantify the sensitivity of the method to model mismatches and leverage this knowledge to account for the estimate uncertainty. Nevertheless, this is not an easy task as there are dependencies across a range of factors, including the specific object geometry, finger design, object placement, and articulation type.

B. State and Model Estimation

We believe that a tighter integration with perception could be extremely fruitful and help to reduce the amount of prior knowledge needed. A perception system could be devised to extract local 3-D patches that are used as candidate interaction hotspots. On the other hand, a surface mesh is not sufficient to model the object motion and dynamical behavior. As a matter of fact, an articulated object is also described by its kinematic parameters, namely, the center and axis of rotation. Last but not least, mass, damping, and friction could be estimated. However, since we did not tune any of these parameters to accurately match the real articulated object in our experiments, we conclude that the method is less sensitive to uncertainty in these model parameters.

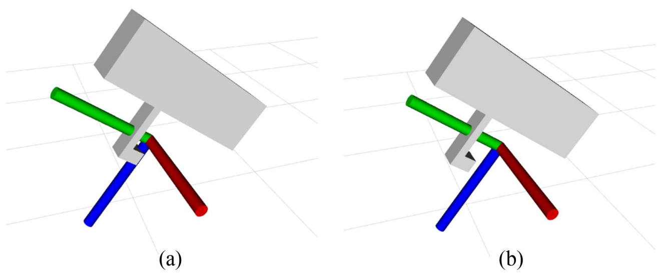


Fig. 14. Location of the gripper frame used to compute the target reaching cost for manipulation can play a big role in finding a successful manipulation strategy. (a) Bad frame placement. (b) Good frame placement.

C. Cost Engineering

The generation of good trajectories strongly depends on a well-engineered and task-dependent cost formulation. With poor cost tuning, stochastic policy optimization, as in the presented method, will tend to become trapped in local minima. Recall that during the manipulation task, we rely mainly on two cost components. We use an end-effector pose cost to drive a virtual gripper frame to the location of an interaction frame, located on the medial axis of the handle. This serves as an interaction location prior. A second door-opening cost instead favors trajectories that pull the door open. If the first term is too high relative to the second, sampling could lead to suboptimal solutions where the end-effector hovers in the proximity of the handle. In fact, pose deviations, from which some are able to successfully pull the door open, are highly penalized.

A second design aspect that influenced sampling quality is the placement of the virtual gripper frame. If this was too close to the finger collision geometry, random sampling would often fail to find a solution that avoids collision and successfully reaches the target pose. Instead, when this is placed at a small offset away from the fingertips, one can prevent early collision and successfully reach the prescribed target pose. Both placements are shown in Fig. 14. While in this work we focus on a practical method for high-rate closed-loop control, a larger sampling budget or a multimodal policy distribution [33] could help to find better solutions but always at the cost of higher compute and lower control rates.

In future work, we would like to alleviate these issues by developing solutions that do not depend on more sampling and rather simplify cost engineering in a way to make it less sensitive to tuning and more generalizable across tasks. One way of achieving this goal would be to use a perception system to extract candidate interaction points and orientations from sensor data (as in [44]), as an automated way of generating an "interaction prior" in a continuous and real-time fashion.

D. Sim-to-Real Gap

Last but not least, particular care must be taken regarding the chosen physics engine used as a rollouts provider. Tuning is often required to reduce the compute time and allow for a longer prediction horizon. One simple way, also adopted in this article, is to increase the simulation step size at the cost of accuracy. This is in part compensated by the closed-loop nature of the control method that will reset the simulation to the newly observed robot state and, therefore, limit the simulation drift. Nowadays, there exists a plethora of physics engines that differ in algorithms, programming language, and functionalities. A quantitative evaluation of the simulation accuracy and a comparative analysis is out of the scope of this article and recent reviews suggest that there is not an obvious winner [45]. Nevertheless, when the task to solve is more complex, including contacts between more bodies and extends over a longer prediction horizon, we think that simulation fidelity can substantially improve the controller performance and, therefore, should be further investigated.

VIII. Conclusion

In this article, we presented a novel control framework for interaction control. We showed its applicability to the real platform and enhanced its robustness adding new algorithmic components, which deploys ZBFs and passivity. We devised a cascaded control architecture that takes into account real-time constraints and for the first time applied these methods on a mobile manipulation platform in real-world experiments. The proposed solution is robust and guarantees stability and safety in the case of unforeseen and sudden safety-critical events as we have demonstrated in our simulated and hardware experiments. Last but not least, we open sourced an efficient multithreaded implementation of the algorithms that we hope can help practitioners to extend and use this method on new applications.

Appendix A
Derivation of Policy Gradient

Recall that the optimization variables are the parameters of the current policy, namely, the mean vector of the normally distributed control sequence $\boldsymbol{\theta} = \{\boldsymbol{\mu}_t, \boldsymbol{\mu}_{t+\Delta t}, \ldots, \boldsymbol{\mu}_{t+T-\Delta t}\}$. With a small abuse of notation, we denote the input command at time index k in the horizon, $\boldsymbol{u}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \Sigma)$. Recall also that we map costs to *success likelihoods* using the exponential function $p(\mathcal{O}|X_t, U_t) = \exp(-\lambda J(X_t, U_t)) = \mathcal{J}_t$. We show the derivation of the gradient for one nominal input vector $\boldsymbol{\mu}_k$

$$\nabla_{\boldsymbol{\mu}_k} \log \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[p(\mathcal{O}|X_t, U_t)\right] = \frac{\nabla_{\boldsymbol{\mu}_k} \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[p(\mathcal{O}|X_t, U_t)\right]}{\mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[p(\mathcal{O}|X_t, U_t)\right]} \tag{59}$$

$$= \frac{\nabla_{\boldsymbol{\mu}_k} \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\mathcal{J}_t\right]}{\mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\mathcal{J}_t\right]}. \tag{60}$$

We further expand the numerator in the previous expression

$$\nabla_{\boldsymbol{\mu}_k} \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\mathcal{J}_t\right] = \nabla_{\boldsymbol{\mu}_k} \int \mathcal{J}_t \pi_{\boldsymbol{\theta}}(U_t) \tag{61}$$

$$= \int \mathcal{J}_t \nabla_{\boldsymbol{\mu}_t} \pi_{\boldsymbol{\theta}}(U_t) \tag{62}$$

$$= \int \mathcal{J}_t \pi_{\boldsymbol{\theta}}(U_t) \nabla_{\boldsymbol{\mu}_k} \log \pi_{\boldsymbol{\theta}}(U_t) \tag{63}$$

$$= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\mathcal{J}_t \nabla_{\boldsymbol{\mu}_k} \log \pi_{\boldsymbol{\theta}}(U_t)] \tag{64}$$

where step (61) follows from the independence of the success likelihood from the policy parameters. We now look at the gradient of the policy log-likelihood with respect to the mean vector

$$\nabla_{\boldsymbol{\mu}_k} \log \pi_{\boldsymbol{\theta}}(U_t) \tag{65}$$

$$= \nabla_{\boldsymbol{\mu}_k} \log \prod_{k'=0}^{K} \frac{1}{\sqrt{(2\pi)^{n_u} |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}||\boldsymbol{u}_{k'} - \boldsymbol{\mu}_{k'}||_{\boldsymbol{\Sigma}^{-1}}\right) \tag{66}$$

$$= \nabla_{\boldsymbol{\mu}_k} \sum_{k'=0}^{K} \log \frac{1}{\sqrt{(2\pi)^{n_u} |\boldsymbol{\Sigma}|}} + \left(-\frac{1}{2}||\boldsymbol{u}_{k'} - \boldsymbol{\mu}_{k'}||_{\boldsymbol{\Sigma}^{-1}}\right) \tag{67}$$

$$= \nabla_{\boldsymbol{\mu}_k} \sum_{k'=0}^{K} -\frac{1}{2}||\boldsymbol{u}_{k'} - \boldsymbol{\mu}_{k'}||_{\boldsymbol{\Sigma}^{-1}} \tag{68}$$

$$= \nabla_{\boldsymbol{\mu}_k} - \frac{1}{2}||\boldsymbol{u}_k - \boldsymbol{\mu}_k||_{\boldsymbol{\Sigma}^{-1}} \tag{69}$$

$$= \boldsymbol{\Sigma}^{-1}(\boldsymbol{u}_k - \boldsymbol{\mu}_k) \tag{70}$$

$$= \boldsymbol{\Sigma}^{-1}\boldsymbol{\varepsilon}_k \tag{71}$$

with K being the total number of discrete-time steps in the time horizon. Plugging the previous expression into (64), we obtain

$$\mathbb{E}_{\boldsymbol{\mu}_k}\left[\mathcal{J}_t \nabla_{\boldsymbol{\mu}_k} \log \pi_{\boldsymbol{\theta}}(U_t)\right] = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\mathcal{J}_t \boldsymbol{\Sigma}^{-1} \boldsymbol{\varepsilon}_k\right] \tag{72}$$

$$= \boldsymbol{\Sigma}^{-1} \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\mathcal{J}_t \boldsymbol{\varepsilon}_k\right]. \tag{73}$$

We finally combine (60) and (73) to get the equation that relates the gradient to the cost and input noise

$$\nabla_{\boldsymbol{\mu}_k} \log \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[p(\mathcal{O}|X_t, U_t)\right] = \boldsymbol{\Sigma}^{-1} \frac{\mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\mathcal{J}_t \boldsymbol{\varepsilon}_k\right]}{\mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\mathcal{J}_t\right]} \tag{74}$$

$$= \boldsymbol{\Sigma}^{-1} \frac{\mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\exp(-\lambda J_t) \boldsymbol{\varepsilon}_k\right]}{\mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\exp(-\lambda J_t)\right]}. \tag{75}$$

Appendix B
Passivity Analysis

The manipulator energy is defined as

$$S_{\text{robot}} = \frac{1}{2}\dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{M}(\boldsymbol{q}) \dot{\tilde{\boldsymbol{q}}} + \frac{1}{2}\tilde{\boldsymbol{q}}^T \boldsymbol{K}_I \tilde{\boldsymbol{q}}. \tag{76}$$

The energy dynamics can be obtained by computing the time derivative of the previous expression. We plug the low-level control law in (53) into (2), and exploiting the fact that $\boldsymbol{M}(\boldsymbol{q}) - 2\boldsymbol{C}_r(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is skew symmetric, we get

$$\begin{aligned}\dot{S}_{\text{robot}} &= \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{M}(\boldsymbol{q}) \ddot{\tilde{\boldsymbol{q}}} + \frac{1}{2}\dot{\tilde{\boldsymbol{q}}}^T \dot{\boldsymbol{M}}(\boldsymbol{q}) \dot{\tilde{\boldsymbol{q}}} + \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{K}_I \tilde{\boldsymbol{q}} \\ &= \dot{\tilde{\boldsymbol{q}}}^T \left[-\boldsymbol{C}_r(\boldsymbol{q}, \dot{\boldsymbol{q}}) \dot{\tilde{\boldsymbol{q}}} - \boldsymbol{K}_D \dot{\tilde{\boldsymbol{q}}} - \boldsymbol{K}_I \int_0^t \dot{\tilde{\boldsymbol{q}}} d\tau + \boldsymbol{\tau}_{\text{ext}}\right] \\ &\quad + \frac{1}{2}\dot{\tilde{\boldsymbol{q}}}^T \dot{\boldsymbol{M}}(\boldsymbol{q}) \dot{\tilde{\boldsymbol{q}}} + \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{K}_I \tilde{\boldsymbol{q}} \\ &= \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{K}_I \tilde{\boldsymbol{q}} + \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{\tau}_{\text{ext}} + \frac{1}{2}\dot{\tilde{\boldsymbol{q}}}^T \left[\dot{\boldsymbol{M}}(\boldsymbol{q}) - 2\boldsymbol{C}_r(\boldsymbol{q}, \dot{\boldsymbol{q}})\right] \dot{\tilde{\boldsymbol{q}}}\end{aligned}$$

$$- \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{K}_D \dot{\tilde{\boldsymbol{q}}} - \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{K}_I \int_0^t \dot{\tilde{\boldsymbol{q}}}\, d\tau$$

$$= \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{K}_I \tilde{\boldsymbol{q}} + \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{\tau}_{\text{ext}} - \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{K}_D \dot{\tilde{\boldsymbol{q}}} - \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{K}_I \int_0^t \dot{\tilde{\boldsymbol{q}}}\, d\tau. \qquad (77)$$

As we compute the desired position integrating the desired velocity over time, it holds that $\tilde{\boldsymbol{q}} = \int_0^t \dot{\tilde{\boldsymbol{q}}}\, d\tau$, obtaining

$$\dot{S}_{\text{robot}} = \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{\tau}_{\text{ext}} - \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{K}_D \dot{\tilde{\boldsymbol{q}}} \leq \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{\tau}_{\text{ext}}. \qquad (78)$$

Appendix C
Table of Parameters

TABLE I
METHOD'S MOST RELEVANT PARAMETERS

Sampling-based Controller	rollouts	40	Δt	0.015s
	T	1s	h	10
	$g_{\max}$	0.2	ρ	1.0
	Caching[6]	30%		
	Σ	$[0.5, 0.5, 0.5, 1.25, \ldots, 1.25] \in \mathbb{R}^{11}$		
Cost Function	$\boldsymbol{W}_t$	$10^2 \cdot I,\ I \in \mathbb{R}^6$	w_γ	10^2
	w_j	10^3	$\boldsymbol{W}_{js}$	10^2
	w_r	10^3	w_{rs}	10^2
	$\boldsymbol{W}_o$	10^2	$p_{\max}$	0
	w_p	10		
Savitzky–Golay	Polynomial order		3	
	Window size		30	
Energy Tank	$S(0)$		$10J$	
	ϵ		$2J$	
FILTER-QP Slack Penalties	Input limits	Hard constraint	Joint limits	10
	Cartesian limits	10^2	Self -collision	10^2
	Passivity	10^2		

Note that the same set of parameters have been used for all tested scenarios in simulation and real-world experiments.

References

[1] S. Cooper, A. Di Fava, C. Vivas, L. Marchionni, and F. Ferro, "ARI: The social assistive robot and companion," in *Proc. 29th IEEE Int. Conf. Robot Hum. Interact. Commun.*, 2020, pp. 745–751.

[2] T. Duckett et al., "Agricultural robotics: The future of robotic agriculture," U.K.-RAS Netw., London, U.K., Tech. Rep. ISSN 2398-4414, 2018.

[3] D. Lattanzi and G. Miller, "Review of robotic infrastructure inspection systems," *J. Infrastructure Syst.*, vol. 23, no. 3, 2017, Art. no. 04017004.

[4] A. Murali, A. Mousavian, C. Eppner, C. Paxton, and D. Fox, "6-DOF grasping for target-driven object manipulation in clutter," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 6232–6238.

[5] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 512–519.

[6] Y. Chebotar et al., "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *Proc. Int. Conf. Robot. Autom.*, 2019, pp. 8973–8979.

[7] L. Peric, M. Brunner, K. Bodie, M. Tognon, and R. Siegwart, "Direct force and pose NMPC with multiple interaction modes for aerial push-and-slide operations," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 131–137.

[8] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optimal Control Appl. Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[9] R. Grandia, F. Farshidian, A. Dosovitskiy, R. Ranftl, and M. Hutter, "Frequency-aware model predictive control," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1517–1524, Apr. 2019.

[10] M. V. Minniti, F. Farshidian, R. Grandia, and M. Hutter, "Whole-body MPC for a dynamically stable mobile manipulator," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3687–3694, Oct. 2019.

[11] J. Buchli, F. Farshidian, A. Winkler, T. Sandy, and M. Giftthaler, "Optimal and learning control for autonomous robots," 2017, *arXiv:1708.09342*.

[12] K. Lee, J. Gibson, and E. A. Theodorou, "Aggressive perception-aware navigation using deep optical flow dynamics and PixelMPC," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1207–1214, Apr. 2020.

[13] I. Abraham, A. Handa, N. Ratliff, K. Lowrey, T. D. Murphey, and D. Fox, "Model-based generalization under parameter uncertainty using path integral control," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2864–2871, Apr. 2020.

[14] G. Williams et al., "Information theoretic MPC using neural network dynamics," in *Proc. 30th Conf. Neural Inf. Process. Syst.*, 2016.

[15] G. Williams et al., "Information theoretic MPC for model-based reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 1714–1721.

[16] J. Rajamäki and P. Hämäläinen, "Augmenting sampling based controllers with machine learning," in *Proc. ACM SIGGRAPH/Eurograph. Symp. Comput. Animation*, 2017, pp. 1–9.

[17] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization-based full body control for the DARPA robotics challenge," *J. field Robot.*, vol. 32, no. 2, pp. 293–312, 2015.

[18] M. Bjelonic et al., "Keep Rollin'—Whole-body motion control and planning for wheeled quadrupedal robots," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 2116–2123, Apr. 2019.

[19] M. V. Minniti, R. Grandia, K. Fäh, F. Farshidian, and M. Hutter, "Model predictive robot-environment interaction control for mobile manipulation tasks," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 1651–1657.

[20] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 3861–3876, Aug. 2017.

[21] E. Cuniato, N. Lawrance, M. Tognon, and R. Siegwart, "Power-based safety layer for aerial vehicles in physical interaction using Lyapunov exponents," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 6774–6781, Jul. 2022.

[22] A. Vahidi and A. Eskandarian, "Research advances in intelligent collision avoidance and adaptive cruise control," *IEEE Trans. Intell. Transp. Syst.*, vol. 4, no. 3, pp. 143–153, Sep. 2003.

[23] T. Gurriet, A. Singletary, J. Reher, L. Ciarletta, E. Feron, and A. Ames, "Towards a framework for realizable safety critical control through active set invariance," in *Proc. ACM/IEEE 9th Int. Conf. Cyber-Phys. Syst.*, 2018, pp. 98–106.

[24] F. Benzi and C. Secchi, "An optimization approach for a robust and flexible control in collaborative applications," *in Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 3575–3581.

[25] J. Choi, F. Castaneda, C. J. Tomlin, and K. Sreenath, "Reinforcement learning for safety-critical control under model uncertainty, using control Lyapunov functions and control barrier functions," 2020, *arXiv:2004.07584*.

[26] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, pp. 3387–3395.

[27] J. C. Willems, "Dissipative dynamical systems—Part I: General theory," *Arch. Rational Mech. Anal.*, vol. 45, no. 5, pp. 321–351, 1972.

[28] F. Ferraguti et al., "An energy tank-based interactive control architecture for autonomous and teleoperated robotic surgery," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1073–1088, Oct. 2015.

[29] C. Schindlbeck and S. Haddadin, "Unified passivity-based Cartesian force/impedance control for rigid and flexible joint robots via task-energy tanks," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 440–447.

[30] E. Shahriari, L. Johannsmeier, and S. Haddadin, "Valve-based virtual energy tanks: A framework to simultaneously passify controls and embed control objectives," in *Proc. Annu. Amer. Control Conf.*, 2018, pp. 3634–3641.

[31] F. Benzi, M. Brunner, M. Tognon, C. Secchi, and R. Siegwart, "Adaptive tank-based control for aerial physical interaction with uncertain dynamic environments using energy-task estimation," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 9129–9136, Oct. 2022.

[32] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3, pp. 229–256, 1992.

[33] A. Lambert, A. Fishman, D. Fox, B. Boots, and F. Ramos, “Stein variational model predictive control,” in *Proc. Conf. Robot Learn.*, 2020, pp. 1278–1297.
[34] S. Levine, “Reinforcement learning and control as probabilistic inference: Tutorial and review,” 2018, *arXiv:1805.00909*.
[35] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, “Control barrier functions: Theory and applications,” in *Proc. 18th Eur. Control Conf.*, 2019, pp. 3420–3431.
[36] J.-L. Blanco, “A tutorial on SE(3) transformation parameterizations and on-manifold optimization,” Univ. Malaga, Málaga, Spain, Tech. Rep. UMA-MAPIR-012010, 2013.
[37] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, “Information-theoretic model predictive control: Theory and applications to autonomous driving,” *IEEE Trans. Robot.*, vol. 34, no. 6, pp. 1603–1622, Dec. 2018.
[38] E. Theodorou, J. Buchli, and S. Schaal, “A generalized path integral control approach to reinforcement learning,” *J. Mach. Learn. Res.*, vol. 11, pp. 3137–3181, 2010.
[39] P. A. Gorry, “General least-squares smoothing and differentiation by the convolution Savitzky-Golay method,” *Anal. Chem.*, vol. 62, no. 6, pp. 570–573, 1990.
[40] T. Erez, Y. Tassa, and E. Todorov, “Simulation tools for model-based robotics: Comparison of Bullet, Havok, MuJoCo, ODE and PhysX,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 4397–4404.
[41] *Articulated Systems: PD Controller*, RaiSim Tech, Zürich, Switzerland, 2022. [Online]. Available: https://raisim.com/sections/ArticulatedSystem.html#pd-controller
[42] *RaiSim: A Cross-Platform Multi-Body Physics Engine for Robotics and AI*, RaiSim Tech, Zürich, Switzerland, 2021. [Online]. Available: https://raisim.com/
[43] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, “OSQP: An operator splitting solver for quadratic programs,” *Math. Program. Comput.*, vol. 12, no. 4, pp. 637–672, 2020.
[44] T. Nagarajan, C. Feichtenhofer, and K. Grauman, “Grounded human-object interaction hotspots from video,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 8688–8697.
[45] J. Collins, S. Chand, A. Vanderkop, and D. Howard, “A review of physics simulators for robotic applications,” *IEEE Access*, vol. 9, pp. 51416–51431, 2021.

Giuseppe Rizzi received the B.Sc. degree in mechanical engineering from the Politecnico di Milano, Milan, Italy, in 2016, and the M.Sc. degree in robotics, systems, and control in 2019 from ETH Zürich, Zürich, Switzerland, where he is currently working toward the Ph.D. degree with the Autonomous System Lab.

In 2019, he joined the Robotic Systems Lab as a Research Assistant and worked on autonomous navigation and manipulation for the MBZIRC competition. In 2020, he started his Ph.D. program and joined the Mobile Manipulation Team. His research interests include control and perception for robotic manipulation.

Jen Jen Chung (Member, IEEE) received the B.E. degree in aeronautical (space) engineering from the University of Sydney, Camperdown, NSW, Australia, in 2010, and the Ph.D. degree in information-based exploration–exploitation strategies for autonomous soaring platforms from the Australian Centre for Field Robotics, University of Sydney in 2014.

She is currently an Associate Professor of Mechatronics with the School of Information Technology and Electrical Engineering, University of Queensland, Brisbane, QLD, Australia. From 2018 to 2022, she was a Senior Researcher with the Autonomous Systems Lab, ETH Zürich, Zürich, Switzerland. From 2014 to 2017, she was a Postdoctoral Scholar with Oregon State University, Corvallis, OR, USA, where she researched multiagent learning methods. Her current research interests include perception, planning, and learning for robotic mobile manipulation, algorithms for robot navigation through human crowds, informative path planning, and adaptive sampling.

Abel Gawel received the Ph.D. degree in robotics and machine learning from ETH Zürich, Zürich, Switzerland, in 2018.

He is currently a Principal Researcher in Computer Vision and Machine Learning with the Huawei European Research Center, Zürich. Before that, he was a Senior Scientist with the Autonomous Systems Lab, ETH Zürich, Zürich. In 2019, he was a Visiting Postdoctoral Fellow with the CRI Group, Nanyang Technological University, Singapore. Prior to joining ETH Zürich in 2014, he was with Bosch Corporate Research and with the BMW Group. His research interests include simultaneous localization and mapping, high-accuracy localization, object recognition, and semantic scene understanding with application in construction robotics, industrial inspection, and search and rescue robotics.

Lionel Ott received the Ph.D. degree in robotics and machine learning from the University of Sydney, Camperdown, NSW, Australia, in 2014.

He is currently a Senior Researcher with the Autonomous Systems Lab (ASL), ETH Zürich, Zürich, Switzerland. Before working with the ASL, he was a Senior Research Fellow with the School of Computer Science, University of Sydney. His research interests include the intersection between machine learning and robotics with the goal to develop methods, techniques, and systems that allow an autonomous agent to build and maintain a representation of the ever-changing world in which robotics systems operate to achieve their given task. His research interests also include model learning and decision making, taking uncertainty into account.

Marco Tognon (Member, IEEE) received the M.Sc. degree in automation engineering from the University of Padua, Padua, Italy, in 2014, with a master thesis carried out at the Max Planck Institute for Biological Cybernetics, Tübingen, Germany, and the Ph.D. degree in robotics from the National Institute of Applied Sciences of Toulouse, Toulouse, France, in 2018, developing his thesis at the Laboratory for Analysis and Architecture of Systems, CNRS (LAAS-CNRS), Toulouse.

Since November 2022, he has been a Tenured Researcher with the Rainbow Team, INRIA Rennes, Rennes, France. Before that, he was a Postdoctoral Researcher with the Autonomous System Lab, ETH Zürich, Zürich, Switzerland. From 2018 to 2020, he was a Postdoctoral Researcher with LAAS-CNRS. His current research interests include robotics, aerial physical interaction, multirobot systems, and human–robot physical interaction.

Dr. Tognon received three prizes for his Ph.D. thesis.

Roland Siegwart (Fellow, IEEE) received the Master in mechanical engineering, Swiss Federal Institute of Technology, ETH Zurich, in 1984, and the Ph.D. in mechatronics engineering, ETH Zurich, with distinction (ETH Medal), in 1989.

He is a Professor of Autonomous Mobile Robots with ETH Zürich, Zürich, Switzerland, a Founding Co-Director of the Technology Transfer Center, Wyss Zurich, Zurich, and a Board Member of multiple high-tech companies. From 1996 to 2006, he was a Professor with the Swiss Federal Institute of Technology Lausanne, Lausanne, Switzerland, held visiting positions with Stanford University, Stanford, CA, USA, and NASA Ames, Mountain View, CA, and was a Vice President with ETH Zürich from 2010 to 2014. He is among the most cited scientist in robots worldwide, Co-Founder of more than half a dozen spin-off companies, and a strong promoter of innovation and entrepreneurship in Switzerland. His research interests include the design, control, and navigation of flying, and wheeled and walking robots operating in complex and highly dynamical environments.

Prof. Siegwart is the recipient of the IEEE RAS Pioneer Award and IEEE RAS Inaba Technical Award.