# A Framework for Interaction Control of Mobile Manipulators

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

***Abstract*—In this work we investigate and deploy stochastic control techniques for the challenging task of mobile manipulation. By their nature, manipulation tasks necessitate environment interactions which require the handling of non-differentiable switching contact dynamics. These dynamics represent a strong limitation for traditional gradient-based optimization methods such as Model Predictive Control (MPC) and Differential Dynamic Programming (DDP). *Sampling-based* techniques alleviate these constraints but do not ensure that the robot will be stable and constraints will not be violated. On the other hand, real-world applications in human environment require safety and robustness to unexpected events. For this reason, we propose a novel framework for safe robotic manipulation of movable objects which 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. The proposed controller enables robust deployment of stochastic control using a conventional CPU. We deploy the algorithm on a 10-DOF mobile manipulator robot. A video of the experiments can be found at https://youtu.be/nd_ggrwHi8g. We make our efficient and multi-threaded implementation of the proposed algorithms available as open-source at https://git.io/JXi4d.**

***Index Terms*—Motion Control of Manipulators, Mobile Manipulation, Optimization and Optimal Control, Manipulation of Articulated Objects.**

## 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 healthcare 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.

This work was supported by the European Union H2020 program under project PILOTING No H2020-ICT-2019-2 871542 and project HERON under grant agreement No 955356.

The authors are with the Autonomous Systems Lab, ETH Zürich, Zürich 8092, Switzerland. {grizzi; chungj; gawela; mtognon; liott; rsiegwart}@ethz.ch

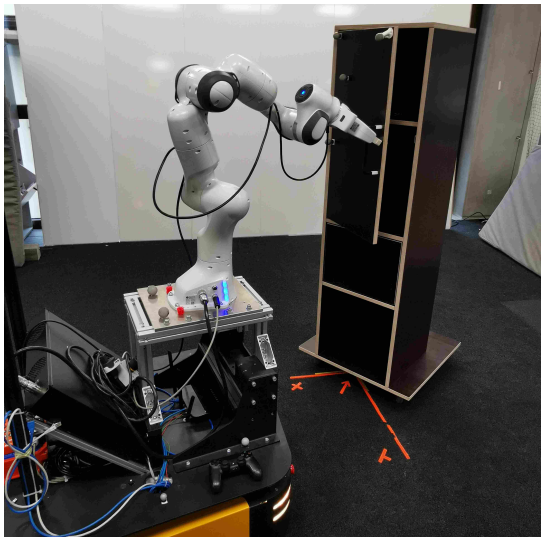


Fig. 1: *RoyalPanda* (a 10-DOF mobile manipulator) while performing a door opening task.

### A. Related Works

While traditional "plan-and-act" frameworks break down manipulation tasks into subproblems that are easier to solve (e.g., reach, grasp, 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]–[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 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 multi-contact task into stages and enforcing constraints when switching between them. This can limit the control envelope of the system and sacrifices solution optimality. For example, 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.

In order to avoid safety-critical configurations additional cost terms are often formulated. However, while these penalize unsafe paths, they do not prohibit them. As a consequence, constraints fulfillment merely relies on sampling "safe" trajectories. Recently, *Control Barrier Functions* (CBFs) have been introduced as a means 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 point-wise in the candidate safe set, imply forward invariance of the set and hence safety [17]. This control method has been successfully deployed in safety-critical applications such as adaptive cruise control and lane keeping [18], segway stabilization [19] and human-robot collaboration [20].

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* [21] 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 [22]. 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 [23], force-impedance control [24], and lately with CBFs for controlling a robotic arm [20].

### B. Contributions

In an effort to propose a practical and efficient method for complex robotic manipulation that at the same time provides theoretical guarantees of stability and safety, our main contribution is a formally proven stable and robust receding horizon algorithm that achieves real-time whole-body control of a mobile manipulator for the complex manipulation of movable and articulated objects. This new framework for mobile manipulation combines tools from stochastic optimization, safety-critical control, and passivity theory. The proposed method is based on an iterative sampling of control trajectories to find the optimal strategy that performs the desired manipulation given the current state. The safety and stability of the system are ensured through a sequential optimization problem that finds the closest policy to the optimized one, which respects constraints and the overall passivity of the autonomous system.

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 7-DOF Franka Emika Panda arm mounted on the holonomic Clearpath Ridgeback base, shown in Fig. 1) for a target reaching and door opening task. An open source implementation of our solution is provided at https://git.io/JXi4d.

In summary, the contributions of the present work are summarized as follows:

1) A review of the main theoretical background introducing receding horizon sampling-based control, the concept of barrier functions and energy tanks;
2) a framework for robust and safe model-based receding horizon control of a mobile manipulator;
3) a study of stability through a passivity analysis of the manipulator;
4) practical insights that enhance the performance of the presented algorithms;
5) a simulation evaluation of the effectiveness of the approach with in-depth analysis of the contributions of each algorithmic component;
6) real hardware experiments showing its applicability on a real platform;
7) an open source implementation of the proposed method including a multi-threaded sampling-based controller and an efficient quadratic program for addressing kinematic and dynamic constraints.

### C. Overview

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

## II. Modeling and Problem Formulation

In this work we consider the challenging task of manipulating articulated objects[1] with a mobile manipulator through *non-prehensile 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,

$$\boldsymbol{x} = [\boldsymbol{q}^\top \ \dot{\boldsymbol{q}}^\top \ \boldsymbol{o}^\top \ \dot{\boldsymbol{o}}^\top]^\top \in \mathbb{R}^{2(n_q+n_o)}. \tag{1}$$

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

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

$\boldsymbol{M}_r \in \mathbb{R}^{n_q \times n_q}$ 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 $\boldsymbol{f}_{ext} \in \mathbb{R}^3$ at the contact point into the efforts at the object and robot joints. Coriolis and gravity terms are denoted as $\boldsymbol{b}_r(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and $\boldsymbol{b}_o(\boldsymbol{o}, \dot{\boldsymbol{o}})$.

The system input $\boldsymbol{u} \in \mathbb{R}^{n_q}$ are the desired robot joint velocities $\dot{\boldsymbol{q}}^*$. The joint torques, denoted by $\boldsymbol{\tau}_{cmd} \in \mathbb{R}^{n_q}$, 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 $t$: $U_t = \{\boldsymbol{u}_t, \boldsymbol{u}_{t+\Delta t}, \dots, \boldsymbol{u}_{t+T-\Delta t}\}$. 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 $X_t = \{\boldsymbol{x}_t, \boldsymbol{x}_{t+\Delta t}, \dots, \boldsymbol{x}_{t+T}\}$ 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 $h : \mathcal{X} \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ 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:

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

[1]We define articulated objects as non-actuated 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 multi-contact case is straightforward.

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:

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

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 non-negative 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 which approximates the tail of the infinite horizon cumulative cost.

## 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 remainder of the paper 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 reinforcement learning [25]. 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* [26] or *Free Energy* [15], here we will refer to *Stochastic Control* (SC) and follow a Bayesian approach similar to [27].

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:

$$p(\mathcal{O} = 1) = \int_{\pi_{\boldsymbol{\theta}}} p(\mathcal{O}|X_t, U_t; \boldsymbol{x}_t) p(X_t, U_t; \boldsymbol{x}_t) \tag{6}$$

$$= \int_{\pi_{\boldsymbol{\theta}}} p(\mathcal{O}|X_t, U_t) p(X_t|U_t) p(U_t) \tag{7}$$

$$= \int_{\pi_{\boldsymbol{\theta}}} p(\mathcal{O}|X_t, U_t) \pi_{\boldsymbol{\theta}}(U_t) \tag{8}$$

$$= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[p(\mathcal{O}|X_t, U_t)]. \tag{9}$$

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:

$$\mathbb{E}_{\pi_{\boldsymbol{\theta}}}[p(\mathcal{O}|X_t, U_t)] \propto \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\mathcal{J}(X_t, U_t)\right] \tag{10}$$

$$= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\exp(-\lambda J(X_t, U_t))\right], \tag{11}$$

where the higher the $\lambda$, 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 $k^{\text{th}}$ mean vector:

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

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:

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

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

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. *Control Barrier Functions* 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 $\mathcal{C}$ [17]. 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,

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

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 feed-forward control is available from a nominal controller, a minimum perturbation on the feed-forward command vector $\boldsymbol{u}_{ff}$ can be found through the following QP [28]:

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

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 turns 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 $\sigma$,

$$S(\boldsymbol{x}(\sigma)) - S(\boldsymbol{x}(0)) \leq \int_0^\sigma \boldsymbol{u}^T \boldsymbol{y}\ dt. \tag{22}$$

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,

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

[3]As shown in [17], CBFs induce an asymptotically stable behavior towards the safe set $\mathcal{C}$. This property is particularly useful when disturbances bring the system outside the constraints.

[4]In comparison, [19] addresses 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.

where $P_{in}$ is the power flowing into the system. As shown in [24], 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, non-passive 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 $S(t) \geq \epsilon > 0, \ \forall t$.

## 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 use the presented sampling method to generate joint velocity command sequences that optimize a task-dependent objective in a receding horizon fashion. We encode a set of tasks and safety-related objectives as different cost terms. Nevertheless, sampling does not inherently provide guarantees and therefore we need a method to enhance safety. We address this problem using an optimization problem as a non-invasive way to change the command sequence when constraints are not fulfilled. In particular, we solve a *sequential* optimization problem (QP), which takes as input a command sequence and the starting state, and returns a modified command-state sequence satisfying all constraints. Nevertheless, as the expensive rollout sampling procedure often limits the rate of the controller, a portion of this input sequence must be executed in *open-loop* which can be unsafe in dynamic environments and for low sampling rates. On the other hand, the QP can be efficiently solved at almost the same rate as the low-level controller. To this end, we propose a *cascaded control* architecture composed of an inner loop where the policy is updated at a low-rate and an outer loop where the low-level controller tracks the policy commands. In the outer loop a QP is solved for the current

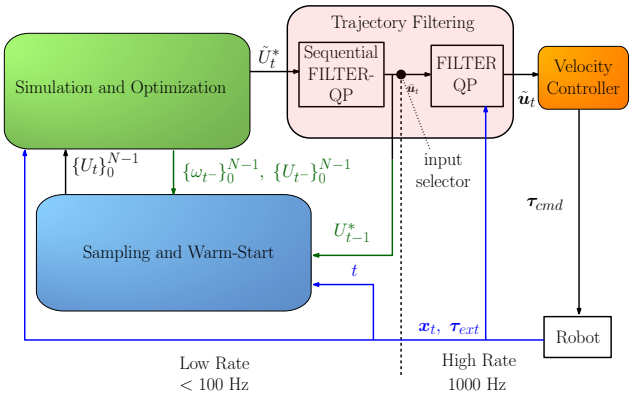



Fig. 2: The figure shows an high-level schematic of the method. The **blue** variables represent the information available at the new optimization step. Some information is reused from the previous iteration to warm start the sampling procedure as shown in **dark green**. The cascaded control scheme is composed by a low-rate input sequence generation and an high-rate velocity command tracking blocks. We propose to modify the input sequence in both blocks through the Sequential FILTER-QP and FILTER-QP as a way to add guarantees to the command sequences generated via sampling and stochastic optimization.

state-input pair in order to find the best input according to the latest received measurements. In the inner loop the same QP is solved for each state-input pair of the optimal rollout trajectory. In other words, the optimization problem is solved:

- *sequentially*, for each command in the optimal input-state trajectory to "sanitize" the sampling-based controller output wherever constraints might be violated,
- *point-wise*, for each command executed by the low-level controller, in order to account for different control rates.

The complete high-level schematic of the full framework is depicted in Fig. 2. Sequential and point-wise optimizations are denoted as Sequential FILTER-QP and FILTER-QP respectively and belong to the *Trajectory Filtering* block of Fig. 2.

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 $\Pi_*$ each controller where $*$ can be $N$, $O$, $I$, $IO$:

$\Pi_N$: relies exclusively on a stochastic controller to generate velocity commands, namely the *Trajectory Filtering* block is completely missing and the sampled input trajectory is tracked as it is by the low-level velocity controller,

$\Pi_O$: a FILTER-QP is used to filter the instantaneous input command to be tracked by the low-level velocity controller,

$\Pi_I$: the Sequential FILTER-QP is used to filter the full optimized command trajectory in a sequential manner but no FILTER-QP is used in the outer control loop,

$\Pi_{IO}$: the full cascaded architecture is deployed, combining the previous two methods.

In the following we present our cost formulation for the manipulation task. This is used in the sampling procedure to weight different trajectory samples and drive the robot to a successful task execution. Then, we encode safety requirements in the form of ZBFs. These allow us to formulate an optimization

problem that ensures forward invariance with respect to the safe set when joint velocities are accurately tracked.

While the safety objectives 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. For this purpose, 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.

### A. Sampling-based Control

The sampling-based framework offers the freedom to directly plan in torque space or use position/velocity control and defer tracking to a low-level controller. In fact, directly planning in the joint space allows us 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). Joint velocity commands are sampled and then translated to motor torques:

$$\boldsymbol{\tau}_{cmd} = \boldsymbol{K}_D(\boldsymbol{u} - \dot{\boldsymbol{q}}) + \boldsymbol{b}_r(\boldsymbol{q}, \dot{\boldsymbol{q}}), \tag{27}$$

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

### B. Cost Shaping

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 $\mathbb{1}[\cdot]$ the *indicator function* such that,

$$\mathbb{1}[x] = \begin{cases} 1 & \text{if } x \text{ is True} \\ 0 & \text{otherwise.} \end{cases} \tag{28}$$

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 non-negative scalar parameters. Note that input constraints are not treated here as additional cost terms as the non-linear dynamics can be augmented with a function that projects the sampled inputs into the feasible set $\mathcal{U}$.

*a) 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 [29]:

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

*b) 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 $\gamma$ can be computed searching for collisions between bodies. $\gamma$ is used to avoid collisions during contact-free motions of the end-effector. The *contact cost* is defined as,

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

*c) 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:

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

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 [30] to help the controller find its way back if poor sampling brings the system outside of the joint position limits.

*d) Arm reach:* we introduce an additional term that penalizes configurations where the arm's end-effector moves excessively relative to the base 2D 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 $r_{\text{curr}} = \boldsymbol{p}^T_{\mathcal{BE}} \boldsymbol{P} \boldsymbol{p}_{\mathcal{BE}}$. The projection matrix $\boldsymbol{P} = \text{diag}(1, 1, 0) \in \mathbb{R}^{3\times3}$ makes sure that the reach is only computed in the 2D plane. Given a maximum reach $r_{\text{max}} \in \mathbb{R}_{\geq 0}$, the *reach cost* is then defined by:

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

*e) 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 $d^{min}_{ij}$ the self-collision cost for the pair is:

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

*f) 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 $\boldsymbol{o}^*$,

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

*g) 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 $c$:

$$\boldsymbol{\tau}_{ext} = \sum_c \boldsymbol{J}_c^T \boldsymbol{f}_c, \tag{35}$$

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

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

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

### C. Cost Scheduling

The manipulation task consists of two phases. In a 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.

### D. Barrier Functions

In the previous subsection a combination of cost components were 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 [20], a simple ZBF can be derived for each joint to keep it between its lower and upper bounds, $q_i^-$ and $q_i^+$, respectively:

$$h_{ql}^i = \frac{(q_i^+ - q)(q - q_i^-)}{(q_i^+ - q_i^-)} \,. \tag{37}$$

In the following we treat the safety requirements associated to 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,

$$\dot{\boldsymbol{p}}_{\mathcal{A}} = \boldsymbol{J}_{\mathcal{A}}^{lin} \dot{\boldsymbol{q}} \,, \tag{38}$$

with $\boldsymbol{J}_{\mathcal{A}}^{lin}$ 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 non-intersecting spheres. Then the self-collision ZBF is defined as,

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

where $d_c = r_i + r_j$ is the sum of the radius of the two collision spheres associated with the $i^{\text{th}}$ and $j^{\text{th}}$ 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,

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

The projection matrix $\boldsymbol{P} = \text{diag}(1, 1, 0) \in \mathbb{R}^{3\times3}$ makes sure that the reach is only computed in the 2D 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 Sec. III, we can formulate a quadratic program to find the command which is the closest to the sampled one while also satisfying the ZBF constraints previously introduced:

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

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. 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 point-wise, 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 Alg. 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.

**Algorithm 1:** Sequential FILTER-QP

**Data:** optimal input sequence $U_t^* = \{\boldsymbol{u}_t^*, \ldots, \boldsymbol{u}_{t+T-\Delta t}^*\} = \{\boldsymbol{\mu}_t, \ldots, \boldsymbol{\mu}_{t+T-\Delta t}\}$ , 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)
  $k = k + 1$
  $\boldsymbol{u} \leftarrow \boldsymbol{u}_{t+k\Delta t}^*$

### E. Energy Tank

While the optimization in (41) enhances the safety of the system we still lack stability guarantees. As described in Sec. III, energy tanks can be used to make the system *passive* and thus ensure stability. Inspired by the work in [20] and [24], this adaptation naturally fits the control method we have developed so far. Let us consider the dynamic manipulator as described in (2). Furthermore, the system uses a dynamically compensated PI velocity controller:

$$\boldsymbol{\tau}_{cmd} = \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}^* + g(\boldsymbol{q}) - \boldsymbol{K}_D\dot{\tilde{\boldsymbol{q}}} - \boldsymbol{K}_I \int_0^\sigma \dot{\tilde{\boldsymbol{q}}}\ dt, \quad (42)$$

with the auxiliary error variable $\tilde{\boldsymbol{q}} = \boldsymbol{q} - \boldsymbol{q}^*$. If we augment the the optimization problem in (41) with the energy constraint:

$$\int_0^\sigma \boldsymbol{\tau}_{ext}^T \dot{\boldsymbol{q}}^*\ dt \geq -S(0), \quad (43)$$

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* (43)*, then the controllers* $\Pi_O$ *and* $\Pi_{IO}$ *make 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,

$$S_{robot} = \frac{1}{2}\dot{\tilde{\boldsymbol{q}}}^T\boldsymbol{M}(\boldsymbol{q})\dot{\tilde{\boldsymbol{q}}} + \frac{1}{2}\tilde{\boldsymbol{q}}^T\boldsymbol{K}_P\tilde{\boldsymbol{q}}. \quad (44)$$

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

$$\dot{S}_{robot} = \dot{\tilde{\boldsymbol{q}}}^T\boldsymbol{\tau}_{ext} - \dot{\tilde{\boldsymbol{q}}}^T\boldsymbol{K}_D\dot{\tilde{\boldsymbol{q}}} \leq \dot{\tilde{\boldsymbol{q}}}^T\boldsymbol{\tau}_{ext}. \quad (45)$$

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 $\dot{S}_{env}$ such that [24],

$$\dot{S}_{env} \leq -\dot{\boldsymbol{q}}^T\boldsymbol{\tau}_{ext}. \quad (46)$$

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

$$\dot{S}_{tank} = \dot{\boldsymbol{q}}^{*T}\boldsymbol{\tau}_{ext}. \quad (47)$$

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

$$\begin{aligned}\dot{S}_{tot} &= \dot{S}_{robot} + \dot{S}_{tank} + \dot{S}_{env} \\ &\leq \dot{\tilde{\boldsymbol{q}}}^T\boldsymbol{\tau}_{ext} + \dot{\boldsymbol{q}}^{*T}\boldsymbol{\tau}_{ext} - \dot{\boldsymbol{q}}^T\boldsymbol{\tau}_{ext} \leq 0,\end{aligned} \quad (48)$$

showing the system's passivity. Intuitively, the increase of the robot's energy is compensated with a reduction of the tank's energy. As the tank energy is limited and bounded from below, passivity is ensured if,

$$S(t) = S(0) + \int_0^\sigma \boldsymbol{\tau}_{ext}^T\dot{\boldsymbol{q}}^*\ dt \geq 0 \quad (49)$$

recovering the constraint in (43) and therefore concluding the proof. □

In practice, the constraint in (43) is formulated as

$$\int_0^\sigma \boldsymbol{\tau}_{ext}^T\boldsymbol{u}\ dt \geq \epsilon - S(0), \quad (50)$$

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

## 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:

$$g_i = \text{sign}(g_i) \cdot \min(|g_i|, g_{max}) \quad (51)$$

where $g_i$ is the i$^{\text{th}}$ 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 $\lambda$ 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 [31] where $\lambda$ is scaled to better discriminate between the experienced trajectories. The modified exponential utility is then defined by,

$$\exp(-\lambda J) = \exp\left(-h\frac{J - J_{\min}}{J_{\max} - J_{\min}}\right), \quad (52)$$

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*),

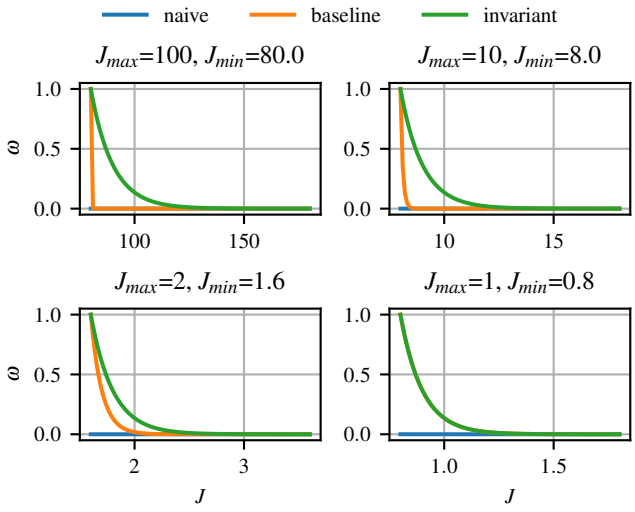



Fig. 3: In each plot we change the maximum and minimum cost (80% of maximum cost). We set $\lambda$ and $h$ to 10 in all comparisons.

with baseline reduction $\mathcal{J} = \exp(-\lambda(J - J_{min}))$ (*baseline*) and the mapping in (52) (*invariant*).

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 which is numerically zero. As a consequence, the gradient estimate in (14) can show a high variance. The mapping in (52) instead, assigns a non-zero 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 $\boldsymbol{u}$:

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

which is equivalent to

$$P_{diss} \leq \alpha E_{res}, \tag{53}$$

where we defined the residual energy $E_{res} = S(x_t(t)) - \epsilon$. 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 $h_{pass} = S(x_t) - \epsilon$ we obtain the same formulation. The ZBF constraint reads as:

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

The passivity can therefore be seen as a ZBF and consequently inherits its properties. A smaller value for $\alpha$ makes the constraint more conservative and improves its performance at the boundary.

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

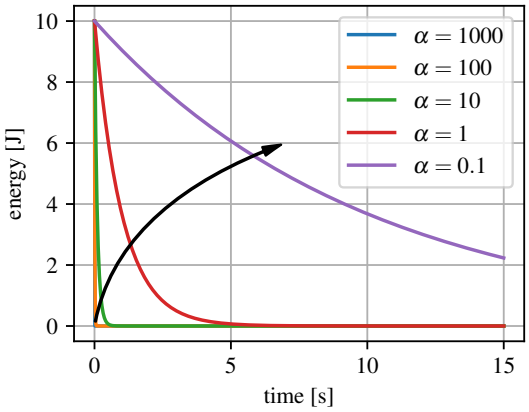



Fig. 4: The figure shows the *worst case* tank energy profile which is how the energy in the tank would evolve if the maximum allowed energy would be dissipated. The smaller the $\alpha$ the more conservative the constraint and thus the energy is depleted at a lower rate.

In order to gain a better intuitive understanding of this tuning parameter, consider the the worst case scenario. In this scenario, the power flow is always equal to the maximal dissipated power allowed by the passivity constraint: $\dot{E}_{res} = -\alpha E_{res}$. Then $E_{res}(t) = E_{res}(0)e^{-\alpha t}$. A smaller $\alpha$ reduces the energy tank deplation 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, re-sampling would perturb it and very likely increase its associated cost. With these motivations in mind, we modify the sampling procedure such that there is always an unperturbed and zero velocity input sequences in the batch of samples.

Furthermore, at the beginning of a new sampling, a subset of the best rollouts are 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 [32] before the Sequential FILTER-QP. The full pipeline is visualized in Fig. 5.

### E. Simulator Tuning

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 over-estimation 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 depends on the

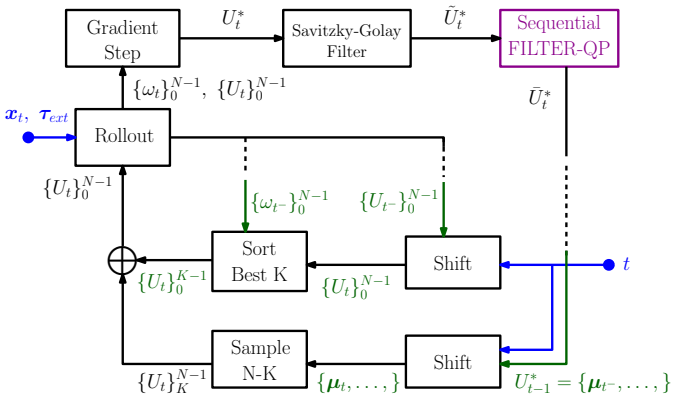
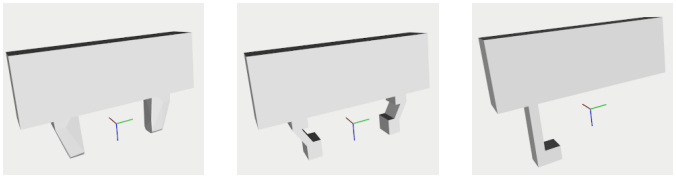



Fig. 5: The figure shows the implementation of the receding horizon scheme. Variables depicted in **blue** and **dark green** represent the information available at the new optimization step and from the previous iteration respectively. All the previous samples and the nominal trajectory are shifted 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 methods $\Pi_I$ and $\Pi_{IO}$.



(a) Original mesh (b) Two fingers (c) Hook finger

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. In contrast, a simplified mesh (shown in (b) and (c)) can bring more accuracy as well as a computational performance gain in terms of simulation rate.

system geometry which can be accurately measured (e.g from CAD models). Therefore solutions that exploit the latter are more likely to succeed on the real platform. One can bias the controller towards these solutions by setting, for example, a very low friction coefficient between contact bodies.

### F. 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 robot end-effector and the object. This adaptation tremendously reduces computation especially during the contact phase. In our following evaluations we observed that using the original mesh shown in Fig. 6a results in a mean simulation rate of 0.98kHz. In contrast, the two finger mesh and hook finger mesh in Fig. 6b-6c allow mean simulation rates of 2.13kHz and 2.94kHz, respectively.

## VI. Numerical and Experimental Evaluation

The goal of this section is to evaluate the proposed control methods. To this end, we define the following metrics:

- *Average stage cost*: this performance metric is computed by averaging the stage cost evaluated at each time step. Task duration and cost scheduling is kept fixed among all the experiments.
- *Cumulative constraints 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:

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

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

$$P_{diss} = \sum_{0}^{T_{exp}} -\boldsymbol{u}^T \boldsymbol{\tau}_{ext}. \tag{54}$$

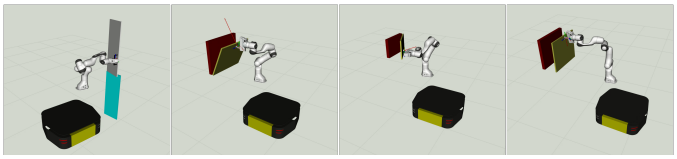


Fig. 7: The four articulated objects used in our simulation evaluations. From left to right: *shelf*, *dishwasher*, *microwave* and *drawer*.

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 an holonomic mobile base equipped with a 7-DOF manipulator. The robot's wrist mounts a 6 axis force-torque sensor and a custom set of fingers as shown in Fig. 6c. The omnidirectional base is controlled by sending velocity commands to the mecanum wheel controller. The arm's low-level controller runs at 1KHz while the base mecanum controller runs at 50Hz. In order to reduce the sim-to-real gap, in all the simulated experiments, imperfect velocity control is modeled by compensating only 90% of the gravity terms. In both the real and simulated robot, the arm velocity commands are converted to joint torques using a PI low-level velocity controller running at 1KHz.

### 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.
2) *Target reaching and obstacle avoidance*: we further investigate the validity of the constraints formulation for

a target reaching task in a confined environment when an obstacle is suddenly placed in the workspace.

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.

We run the presented algorithms on a Intel Core i7-8550U quad-core processor (1.8 GHz, up to 4.0 GHz) and use 8 threads for parallel forward sampling of rollouts. Forward simulation is provided by the `raisim` physics engine [33]. The FILTER-QP and Sequential FILTER-QP are solved efficiently using the `osqp` C++ library [34]. We measure an average solving time of $\approx 0.1$ ms. Finally, the main control parameters are summarized in Tab. I.

*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. $\Pi_I$ and $\Pi_{IO}$ 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 third row of Fig. 8 also shows that the naive controller $\Pi_N$ 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 non-passive behavior on the real robot.

Overall, the simulated experiments confirm the validity of the full framework (as implemented in $\Pi_{IO}$). Nevertheless the performance differences are not striking. We think that this

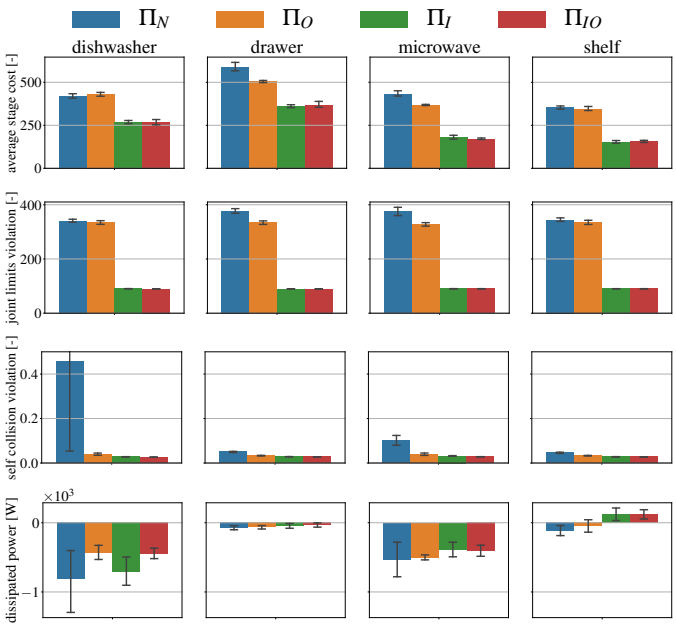


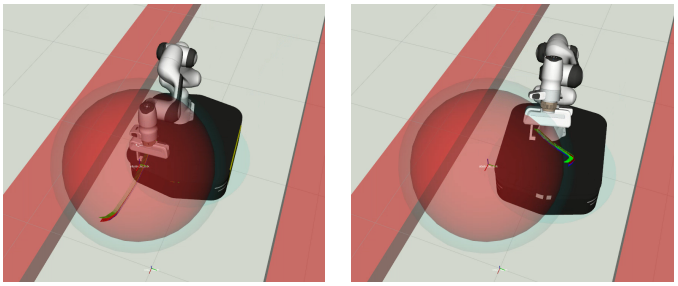

Fig. 8: Comparison between the different control methods. Note that The self-collision barplot has a data point at 3.4 in the $\Pi_N$ method for the *dishwasher* case, which is far outside the plot range.

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 post-processing 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:

- the cost needs to be nicely tuned in order to prevent edge cases where performance is chosen in lieu of safety,
- 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 ($<$ 1cm) causing a sudden change in the cost function. We show the end-effector optimal trajectory for $\Pi_N$ and $\Pi_{IO}$ after the obstacle has been detected in Fig. 9. We can see that the naive controller $\Pi_N$ 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 trade-off



(a) Example rollout of $\Pi_N$ (b) Example rollout of $\Pi_{IO}$

Fig. 9: In the figure we visualize the optimized end-effector trajectory by rolling out the optimized command. The naive controller $\Pi_N$ is trapped in a high cost region while $\Pi_{IO}$ immediately reacts to the unexpected cost change.

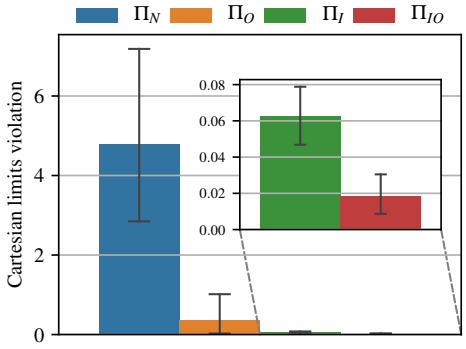


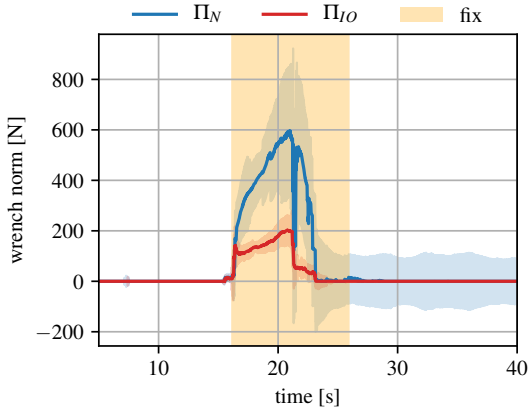

Fig. 10: The barplot shows the cumulative cartesian limits violation for the target reaching and obstacle avoidance task. $\Pi_I$ already outperforms $\Pi_N$ and $\Pi_O$. A further small performance boost is achieved by $\Pi_{IO}$.

between obstacle avoidance and the target reaching objective. In contrast, the $\Pi_{IO}$ 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.

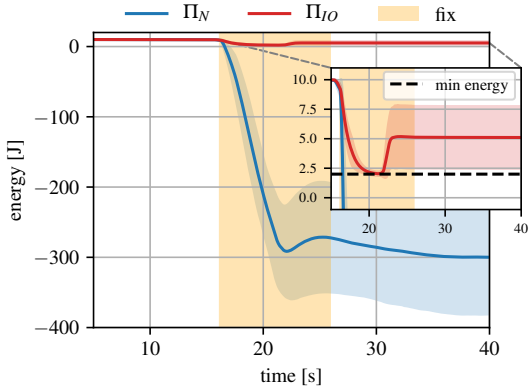
We repeat the simulation for 10 runs for each of the methods. The results are summarized in Fig. 10. As expected, $\Pi_{IO}$ attains the best performance. We further observe that the presence of a FILTER-QP in the outer high-rate loop additionally improves robustness as it enforces constraints during 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 seconds, 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 joints' wear.




(a) The **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) The 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. 11: The 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.

From the results in Fig. 11, we note that when using $\Pi_N$, 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 $\Pi_{IO}$, 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 passivity constraint formulation (53). As discussed in Sec. V and shown in Fig. 12, this choice allows for a smoother power regulation when approaching the lower energy bound when compared to larger values of $\alpha$ and prevents chattering.

*4) Real-world experiments:* The goal of the hardware experiments is to demonstrate that the algorithm can be deployed on a real platform in a real-time setup. For this purpose, we perform a door opening experiment similar to the description provided in the previous section with our RoyalPanda robot. 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. The door motion is limited using a rope. We monitor the evolution of the tank energy and once it has reached the

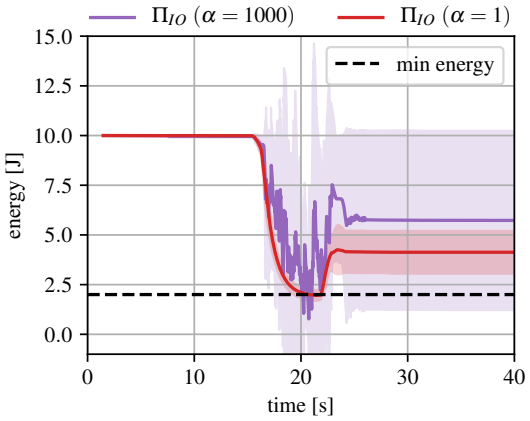



Fig. 12: In this figure we show the chattering effect happening when the constraints are enforced with $\alpha = 1/dt = 1000$. In contrast, a lower $\alpha$ greatly alleviates this issue.

allowed minimum we wait 3 seconds before manually cutting the rope. We repeat the experiment for the methods $\Pi_N$ and $\Pi_{IO}$ recalling that the latter only uses the passivity constraint in the outer loop.

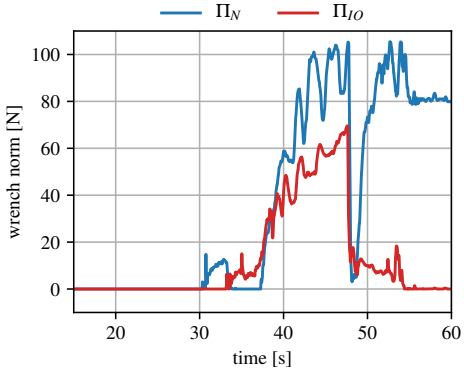
The wrench norm and the evolution of the energy in the tank during the experiments can be seen in Fig. 13. As one can see in the accompanying video, without passivity, the manipulator exerts a rapidly increasing wrench until the rope is broken apart. On the other hand, when passivity is enforced, the 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 $\Pi_N$ is deployed, the controller tries to push aggressively and is therefore not able to escape this gripper trap. When using $\Pi_{IO}$ instead, 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.

## VII. Method Limitations and Future Work

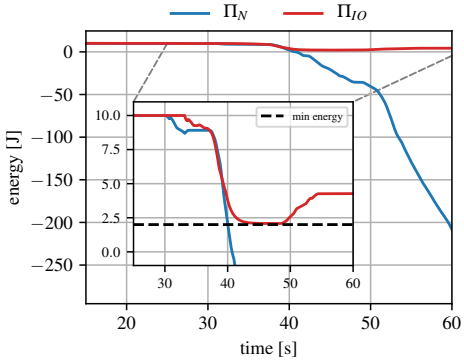
In the following we describe the most prominent issues that emerged during the experiments and how they could be addressed to improve the proposed method.

### 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, the controller would generate a "tapping" motion which would cause a hard collision between the finger tip 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.




(a) Wrench norm during the door opening experiment. We measure an average wrench of 68 N and 34 N for the $\Pi_N$ and $\Pi_{IO}$ 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.

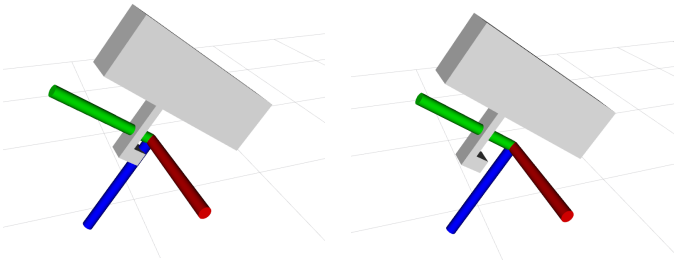
Fig. 13: The figures show that when passivity is enforced, the controller is able to limit the overall interaction wrench. This experiment is also reported in the accompanying video.

### 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 3D 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.

### 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



(a) Bad frame placement. (b) Good frame placement.

Fig. 14: The 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.

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 sub-optimal 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 which 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 finger tips 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 [26] 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 don't 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 [35]), 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 work, 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 work and recent reviews suggest that there isn't an obvious winner [36]. 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. Conclusions

In this work, we have presented a novel control framework for interaction control. We have shown 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 which takes into account real-time constraints and applied for the first time these methods on a mobile manipulation platform in real-world experiments. The proposed solution is robust and guarantees stability and safety in case of unforeseen and sudden safety-critical events as we have demonstrated in our simulated and hardware experiments. Last but not least, we open source an efficient multi-threaded implementation of the algorithms that we hope can help practitioners to extend and use this method on new applications.

## References


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

[2] T. Duckett, S. Pearson, S. Blackmore, B. Grieve, W.-H. Chen, G. Cielniak, J. Cleaversmith, J. Dai, S. Davis, C. Fox, *et al.*, "Agricultural robotics: The future of robotic agriculture," UK-RAS Network, Tech. Rep., 2018.

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

[4] A. Murali, A. Mousavian, C. Eppner, C. Paxton, and D. Fox, "6-DOF grasping for target-driven object manipulation in clutter," in *2020 IEEE Int. Conf. on Robot. and Autom.*, 5 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 *2016 IEEE Int. Conf. on Robot. and Autom.*, 2016, pp. 512–519.

[6] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *2019 Int. Conf. on Robot. and 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 *International Conference on Robotics and Automation (ICRA 2021)*, 2021.

[8] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optimal Control Appl. and 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. and Autom. Lett.*, vol. 4, no. 2, pp. 1517–1524, 2019.

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

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

[12] K. Lee, J. Gibson, and E. A. Theodorou, "Aggressive perception-aware navigation using deep optical flow dynamics and PixelMPC," *IEEE Robot. and Autom. Lett.*, vol. 5, no. 2, 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. and Autom. Lett.*, vol. 5, no. 2, pp. 2864–2871, 2020.

[14] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic MPC using neural network dynamics," in *30th Conf. on Neural Inform. Process. Syst.*, 2016.

[15] ——, "Information theoretic MPC for model-based reinforcement learning," in *2017 IEEE Int. Conf. on Robot. and Autom.*, 2017, pp. 1714–1721.

[16] J. Rajamäki and P. Hämäläinen, "Augmenting sampling based controllers with machine learning," in *Proc. of the ACM SIGGRAPH / Eurographics Symp. on Comput. Animation*, 2017, pp. 1–9.
[17] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.
[18] A. Vahidi and A. Eskandarian, "Research advances in intelligent collision avoidance and adaptive cruise control," *IEEE transactions on intelligent transportation systems*, vol. 4, no. 3, pp. 143–153, 2003.
[19] 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 *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2018, pp. 98–106.
[20] F. Benzi and C. Secchi, "An optimization approach for a robust and flexible control in collaborative applications," *arXiv preprint arXiv:2103.03082*, 2021.
[21] J. C. Willems, "Dissipative dynamical systems part i: General theory," *Archive for rational mechanics and analysis*, vol. 45, no. 5, pp. 321–351, 1972.
[22] F. Ferraguti, N. Preda, A. Manurung, M. Bonfe, O. Lambercy, R. Gassert, R. Muradore, P. Fiorini, and C. Secchi, "An energy tank-based interactive control architecture for autonomous and teleoperated robotic surgery," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1073–1088, 2015.
[23] C. Schindlbeck and S. Haddadin, "Unified passivity-based cartesian force/impedance control for rigid and flexible joint robots via task-energy tanks," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 440–447.
[24] E. Shahriari, L. Johannsmeier, and S. Haddadin, "Valve-based virtual energy tanks: A framework to simultaneously passify controls and embed control objectives," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 3634–3641.
[25] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3, pp. 229–256, 1992.
[26] A. Lambert, A. Fishman, D. Fox, B. Boots, and F. Ramos, "Stein variational model predictive control," in *Conf. on Robot Learn.*, 2020.
[27] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," *arXiv preprint arXiv:1805.00909*, 2018.
[28] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 3420–3431.
[29] J.-L. Blanco, "A tutorial on SE(3) transformation parameterizations and on-manifold optimization," University of Malaga, Tech. Rep., 2013.
[30] 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. on Robot.*, vol. 34, no. 6, pp. 1603–1622, 2018.
[31] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *J. of Mach. Learn. Res.*, vol. 11, pp. 3137–3181, 2010.
[32] P. A. Gorry, "General least-squares smoothing and differentiation by the convolution Savitzky-Golay method," *Analytical Chemistry*, vol. 62, no. 6, pp. 570–573, 1990.
[33] R. Tech. (2021) Raisim: a cross-platform multi-body physics engine for robotics and ai. [Online]. Available: https://raisim.com/
[34] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020. [Online]. Available: https://doi.org/10.1007/s12532-020-00179-2
[35] T. Nagarajan, C. Feichtenhofer, and K. Grauman, "Grounded human-object interaction hotspots from video," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8688–8697.
[36] J. Collins, S. Chand, A. Vanderkop, and D. Howard, "A review of physics simulators for robotic applications," *IEEE Access*, 2021.

## 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 $\boldsymbol{\theta} = \{\boldsymbol{\mu}_t, \boldsymbol{\mu}_{t+\Delta t}, \dots, \boldsymbol{\mu}_{t+T-\Delta t}\}$. 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 $\boldsymbol{\mu}_k$:

$$\nabla_{\boldsymbol{\mu}_k} \log \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[p(\mathcal{O}|X_t, U_t)\right] = \frac{\nabla_{\boldsymbol{\mu}_k} \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[p(\mathcal{O}|X_t, U_t)\right]}{\mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[p(\mathcal{O}|X_t, U_t)\right]} \tag{55}$$

$$= \frac{\nabla_{\boldsymbol{\mu}_k} \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[\mathcal{J}_t\right]}{\mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[\mathcal{J}_t\right]}. \tag{56}$$

We further expand the numerator in the previous expression:

$$\nabla_{\boldsymbol{\mu}_k} \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[\mathcal{J}_t\right] = \nabla_{\boldsymbol{\mu}_k} \int \mathcal{J}_t \pi_{\boldsymbol{\theta}}(U_t) \tag{57}$$

$$= \int \mathcal{J}_t \nabla_{\boldsymbol{\mu}_t} \pi_{\boldsymbol{\theta}}(U_t) \tag{58}$$

$$= \int \mathcal{J}_t \pi_{\boldsymbol{\theta}}(U_t) \nabla_{\boldsymbol{\mu}_k} \log \pi_{\boldsymbol{\theta}}(U_t) \tag{59}$$

$$= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\mathcal{J}_t \nabla_{\boldsymbol{\mu}_k} \log \pi_{\boldsymbol{\theta}}(U_t)], \tag{60}$$

where step (57) 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:

$$\nabla_{\boldsymbol{\mu}_k} \log \pi_{\boldsymbol{\theta}}(U_t) \tag{61}$$

$$= \nabla_{\boldsymbol{\mu}_k} \log \prod_{k'=0}^{K} \frac{1}{\sqrt{(2\pi)^{n_u} |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2} ||\boldsymbol{u}_{k'} - \boldsymbol{\mu}_{k'}||_{\boldsymbol{\Sigma}^{-1}}\right) \tag{62}$$

$$= \nabla_{\boldsymbol{\mu}_k} \sum_{k'=0}^{K} \log \frac{1}{\sqrt{(2\pi)^{n_u} |\boldsymbol{\Sigma}|}} + \left(-\frac{1}{2} ||\boldsymbol{u}_{k'} - \boldsymbol{\mu}_{k'}||_{\boldsymbol{\Sigma}^{-1}}\right) \tag{63}$$

$$= \nabla_{\boldsymbol{\mu}_k} \sum_{k'=0}^{K} -\frac{1}{2} ||\boldsymbol{u}_{k'} - \boldsymbol{\mu}_{k'}||_{\boldsymbol{\Sigma}^{-1}} \tag{64}$$

$$= \nabla_{\boldsymbol{\mu}_k} -\frac{1}{2} ||\boldsymbol{u}_k - \boldsymbol{\mu}_k||_{\boldsymbol{\Sigma}^{-1}} \tag{65}$$

$$= \boldsymbol{\Sigma}^{-1}(\boldsymbol{u}_k - \boldsymbol{\mu}_k) \tag{66}$$

$$= \boldsymbol{\Sigma}^{-1} \boldsymbol{\varepsilon}_k \tag{67}$$

with $K$ being the total number of discrete time steps in the time horizon. Plugging the previous expression into (60) we obtain:

$$\mathbb{E}_{\boldsymbol{\mu}_k}[\mathcal{J}_t \nabla_{\boldsymbol{\mu}_k} \log \pi_{\boldsymbol{\theta}}(U_t)] = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\mathcal{J}_t \boldsymbol{\Sigma}^{-1} \boldsymbol{\varepsilon}_k] \tag{68}$$

$$= \boldsymbol{\Sigma}^{-1} \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[\mathcal{J}_t \boldsymbol{\varepsilon}_k\right]. \tag{69}$$

We finally combine (69) and (56) to get the equation which relates the gradient to the cost and input noise:

$$\nabla_{\boldsymbol{\mu}_k} \log \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[p(\mathcal{O}|X_t, U_t)\right] = \boldsymbol{\Sigma}^{-1} \frac{\mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[\mathcal{J}_t \boldsymbol{\varepsilon}_k\right]}{\mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[\mathcal{J}_t\right]} \tag{70}$$

$$= \boldsymbol{\Sigma}^{-1} \frac{\mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[\exp(-\lambda J_t) \boldsymbol{\varepsilon}_k\right]}{\mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[\exp(-\lambda J_t)\right]}. \tag{71}$$

## APPENDIX B
## PASSIVITY ANALYSIS

The manipulator energy is defined as:

$$S_{robot} = \frac{1}{2} \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{M}(\boldsymbol{q}) \dot{\tilde{\boldsymbol{q}}} + \frac{1}{2} \tilde{\boldsymbol{q}}^T \boldsymbol{K}_P \tilde{\boldsymbol{q}}. \tag{72}$$

The energy dynamics can be obtained by computing the time derivative of the previous expression. We plug the low level control law in (42) into (2) and exploiting the fact that $\boldsymbol{M}(\boldsymbol{q}) - 2\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is skew symmetric we get:

$$\begin{aligned}
\dot{S}_{robot} &= \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{M}(\boldsymbol{q})\ddot{\tilde{\boldsymbol{q}}} + \frac{1}{2}\dot{\tilde{\boldsymbol{q}}}^T \dot{\boldsymbol{M}}(\boldsymbol{q})\dot{\tilde{\boldsymbol{q}}} + \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{K}_P \tilde{\boldsymbol{q}} \\
&= \dot{\tilde{\boldsymbol{q}}}^T \left[ -\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\tilde{\boldsymbol{q}}} - \boldsymbol{K}_D \dot{\tilde{\boldsymbol{q}}} - \boldsymbol{K}_I \int_0^\sigma \dot{\tilde{\boldsymbol{q}}} dt + \boldsymbol{\tau}_{ext} \right] \\
&\quad + \frac{1}{2}\dot{\tilde{\boldsymbol{q}}}^T \dot{\boldsymbol{M}}(\boldsymbol{q})\dot{\tilde{\boldsymbol{q}}} + \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{K}_P \tilde{\boldsymbol{q}} \\
&= \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{K}_P \tilde{\boldsymbol{q}} + \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{\tau}_{ext} + \frac{1}{2}\dot{\tilde{\boldsymbol{q}}}^T \left[ \dot{\boldsymbol{M}}(\boldsymbol{q}) - 2\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \right] \dot{\tilde{\boldsymbol{q}}} \\
&\quad - \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{K}_D \dot{\tilde{\boldsymbol{q}}} - \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{K}_I \int_0^\sigma \dot{\tilde{\boldsymbol{q}}}\ dt \\
&= \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{K}_P \tilde{\boldsymbol{q}} + \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{\tau}_{ext} - \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{K}_D \dot{\tilde{\boldsymbol{q}}} - \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{K}_I \int_0^\sigma \dot{\tilde{\boldsymbol{q}}}\ dt.
\end{aligned} \tag{73}$$

As we compute the desired position integrating the desired velocity over time, it holds that $\tilde{\boldsymbol{q}} = \int_0^\sigma \dot{\tilde{\boldsymbol{q}}}\ dt$. We choose $\boldsymbol{K}_P = \boldsymbol{K}_I$, obtaining,

$$\dot{S}_{robot} = \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{\tau}_{ext} - \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{K}_D \dot{\tilde{\boldsymbol{q}}} \leq \dot{\tilde{\boldsymbol{q}}}^T \boldsymbol{\tau}_{ext}. \tag{74}$$

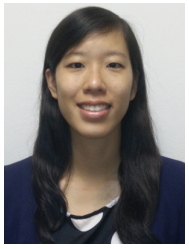
## APPENDIX C
## TABLE OF PARAMETERS

| | | | | |
|---|---|---|---|---|
| Sampling-based Controller | rollouts | 40 | $\Delta t$ | 0.015s |
| | $T$ | 1s | $h$ | 10 |
| | $g_{max}$ | 0.2 | $\rho$ | 1.0 |
| | caching[6] | 30% | | |
| | $\Sigma$ | $[0.5, 0.5, 0.5, 1.25, \ldots, 1.25] \in \mathbb{R}^{11}$ | | |
| Cost Function | $\boldsymbol{W}_t$ | $10^2 \cdot I,\ I \in \mathbb{R}^6$ | $w_\gamma$ | $10^2$ |
| | $w_j$ | $10^3$ | $\boldsymbol{W}_{js}$ | $10^2$ |
| | $w_r$ | $10^3$ | $w_{rs}$ | $10^2$ |
| | $\boldsymbol{W}_o$ | $10^2$ | $p_{max}$ | 0 |
| | $w_p$ | 10 | | |
| Savitzky-Golay | polynomial order | | 3 | |
| | window size | | 30 | |
| Energy Tank | $S(0)$ | | $10J$ | |
| | $\epsilon$ | | $2J$ | |
| FILTER-QP Slack Penalties | input limits | hard constraint | joint limits | 10 |
| | cartesian limits | $10^2$ | self collision | $10^2$ |
| | passivity | $10^2$ | | |

TABLE I: The table summarizes the method's most relevant parameters. Note that the same set of parameters have been used for all tested scenarios in simulation and real-world experiments.

**Giuseppe Rizzi** is a PhD student in the Autonomous System Lab at ETH Zürich. He received a B.Sc in Mechanical Engineering at Politecnico di Milano in 2016 and a M.Sc in Robotics, Systems and Control at ETH Zürich in 2019. He joined the Robotic Systems Lab (RSL) as a Research Assistant and worked on autonomous navigation and manipulation for the MBZIRC competition in 2019. In 2020 he started is PhD program and joined the Mobile Manipulation Team. His research interests include control and perception for robotic manipulation.

[6]Portion of rollouts reused from the previous iteration



**Jen Jen Chung** (Member, IEEE) is a Senior Researcher in the Autonomous Systems Lab (ASL) at ETH Zürich. Her current research interests include perception and learning for mobile manipulation, algorithms for robot navigation through crowds, informative path planning and adaptive sampling. Prior to working at ASL, Jen Jen was a postdoctoral scholar at Oregon State University researching multiagent learning methods and she completed her Ph.D. on information-based exploration-exploitation strategies for autonomous soaring platforms at the Australian Centre for Field Robotics in the University of Sydney. She received her Ph.D. (2014) and B.E. (2010) from the University of Sydney.

**Abel Gawel** is currently a Senior Researcher in Computer Vision and Machine Learning with the Huawei European Research Center. Before that he was a Senior Scientist at the Autonomous Systems Lab of ETH Zurich. He received the PhD from ETH Zurich in 2018 and was a visiting Postdoctoral Fellow in the CRI group at NTU Singapore in 2019. His research interests include SLAM, high-accuracy localization, object recognition, and semantic scene understanding with application in construction robotics, industrial inspection, and search and rescue robotics. Prior to joining ETH in 2014, he worked for Bosch Corporate Research and the BMW group.

**Marco Tognon** (S'15-M'18) is a postdoctoral researcher in the Autonomous System Lab (ASL) at ETH Zurich, Switzerland. He received the Ph.D. degree in Robotics in 2018, from INSA Toulouse, France, developing his thesis at LAAS-CNRS. His thesis has been awarded with three prizes. He received the M.Sc. degree in automation engineering in 2014, from the University of Padua, Italy, with a master thesis carried out at MPI for Biological Cybernetics, Tübingen, Germany. From 2018 to 2020 he has been postdoctoral researcher at LAAS-CNRS, Toulouse, France. His current research interests include robotics, aerial physical interaction, multi-robot systems, and human-robot physical interaction.

**Lionel Ott** is a Senior Researcher in the Autonomous Systems Lab (ASL) at ETH Zurich. His research focuses on 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 interests are in model learning and decision making, taking uncertainty into account. Before working at ASL, Lionel was a senior research fellow in the School of Computer Science at the University of Sydney. He received his Ph.D. in 2014 from the University of Sydney.

**Roland Siegwart** (Fellow, IEEE) is a Professor for autonomous mobile robots with ETH Zurich, Founding Co-Director of the Technology Transfer Center, Wyss Zurich and Board Member of multiple high-tech companies. From 1996 to 2006, he was a Professor with EPFL Lausanne, held visiting positions with Stanford University and NASA Ames and was Vice President of ETH Zurich from 2010 to 2014. His research interest include the design, control, and navigation of flying, and wheeled and walking robots operating in complex and highly dynamical environments.

Prof. Siegwart received the IEEE RAS Pioneer Award and IEEE RAS Inaba Technical Award. 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.