Stochastic Control for Reactive Whole-Body Manipulation

Giuseppe Rizzi, Jen Jen Chung, Abel Gawel, Marco Tognon, Lionel Ott and Roland Siegwart

***Abstract*— In this work we investigate and deploy stochastic control techniques for the challenging task of mobile manipulation. Manipulation tasks require a high degree of interaction. The non-differentiability of the switching contact dynamics represents a strong limitation for traditional gradient-based optimization methods such as MPC and DDP. Sampling-based techniques alleviate these constraints but have been mainly applied in simulation or for low-dimensional dynamical systems. In this letter, we review and adopt sampling methods for solving an object manipulation task with a mobile manipulator and propose adaptations that enable real-time deployment of stochastic control. We evaluate the method on whole-body coordination and manipulation tasks. The method achieves real-time control of a 10-DOF mobile manipulator robot on a conventional CPU. A video of the experiments can be found at `https://youtu.be/4mTHYehNMCc`.**

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]. 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.

In this work, we focus on whole-body coordination and manipulation of articulated objects, e.g., to open drawers or doors (see Fig. 1). These tasks involve a high degree of physical interaction, which necessitates a control strategy that is able to exploit contacts to successfully manipulate the articulated object while satisfying joint and input constraints. While traditional "plan-and-act" frameworks break down such tasks into subproblems that are easier to solve (e.g. reach, grasp, pull) [4], they do not offer fast and control-aware replanning, which is crucial for mobile manipulation operations in dynamic and uncertain environments. We observe that humans heavily rely on their physical understanding of the surrounding environment and thus we believe that a robot needs to consider this aspect as well when planning for the optimal action.

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 capability of dealing 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.

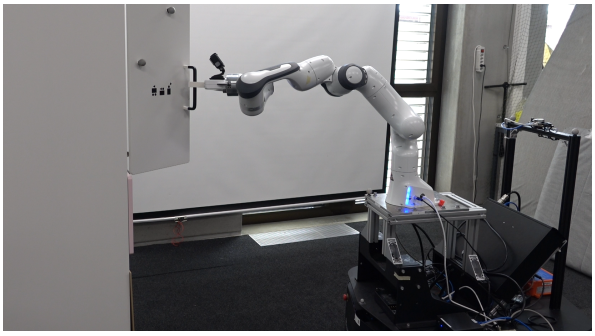


Fig. 1: *RoyalPanda*, a 10-DOF mobile manipulator while performing a door opening task.

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 big optimization, 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].

Yet despite their appealing features, sampling-based methods can be costly to execute and solution quality is highly dependent on sampling quality. For this reason, much of the work in this field has investigated information-theoretic approaches to enable better sampling of the simulation rollouts. Previous works argue that thousands of trajectories

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

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

need to be sampled in real time for the effectiveness of the proposed sampling-based algorithm and therefore GPU-based simulation is needed for fast parallel computation [17]. Unfortunately, GPUs are not common on mobile robots (e.g., because of limited power and payload) and often the overall computation times are still not adequate for feedback control. Furthermore, demonstrated solutions for tasks involving different physical interactions (e.g. a robot arm opening a drawer [13]) have been shown on a real system only 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 that might damage the robot as well as the manipulated object. Ultimately, because of the lack of practical implemented solutions, sampling-based control methods are yet to be applied on real mobile manipulators for whole-body control of complex multi-contact tasks.

We propose a practical receding horizon algorithm that achieves real-time whole-body control of a mobile manipulator using only a laptop CPU. Our work is based on Model Predictive Path Integral control (MPPI) [15], which we use as our underlying sampling-based controller for whole-body coordination and control. We incorporate several algorithmic elements that enable our controller to achieve performant results, often requiring fewer than 100 rollouts. Our main contribution is a regularization of our sampling exploration by means of a momentum update. This is inspired by the Bayesian inference viewpoint of [18], which showed that the choice of exploration noise has the effect of tuning the gradient step size of the path integral update. The benefits of this sampling scheme are twofold, it can aid in escaping local minima while also damping strong oscillations in the optimal policy.

We show that our algorithm is able to control a complex system in real-time without the need for massive parallel computation. To demonstrate the applicability and effectiveness of this approach, we perform several ablation studies in simulation on kinematic and dynamical manipulators and deploy the full algorithm on our *RoyalPanda* platform (a 7-DOF Franka Emika Panda arm mounted on the holonomic Clearpath Ridgeback base, Fig. 1) for a target reaching and door opening task. An open source implementation of our solution is provided at `https://git.io/Jtda7`.

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 system input $\boldsymbol{u} \in \mathbb{R}^{n_q}$ are the desired robot joint velocities. 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 described by the following equation of motion:

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

where $\boldsymbol{M}_r \in \mathbb{R}^{n_q \times n_q}$ 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. $\boldsymbol{J}_r$ and $\boldsymbol{J}_o$ map the interaction force $\boldsymbol{f}_{ext} \in \mathbb{R}^3$ at the contact point into the efforts at the object and robot joints. The joint torques, denoted by $\boldsymbol{\tau}$, are computed by a velocity low-level controller as a function of the velocity references $\boldsymbol{u}$. Finally, $\boldsymbol{b}_r(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and $\boldsymbol{b}_o(\boldsymbol{o}, \dot{\boldsymbol{o}})$ gather the non-linearities, Coriolis and gravity terms. In addition, the system must satisfy state and input constraints at every time step. These may arise from hardware limitations (e.g. joint limits) or simply for safety reasons (e.g. maximum velocity at the joints). The robot is equipped with a joint-level controller which maps desired joint velocities $\boldsymbol{u}$ to low-level commands. We define with $\mathcal{X} \subseteq \mathbb{R}^{2(n_q+n_o)}$ and $\mathcal{U} \subseteq \mathbb{R}^{n_q}$ the spaces of admissible states and inputs, respectively. The control objective is to minimize the distance from an initial to final desired configuration through a feedback policy $\pi(x) : \mathcal{X} \to \mathcal{U}$. The *distance* to the goal configuration must be defined for the specific problem and can be computed in a space different to $\mathcal{X}$ via an opportune mapping $h : \mathcal{X} \to \mathcal{H}$:

$$\begin{aligned} \pi^* = \arg\min_{\pi} \lim_{t\to\infty} ||h(\boldsymbol{x}^*) - h(\boldsymbol{x}(t))|| \\ \text{s.t.} \, \forall \, t \in \mathbb{R}_{\geq 0} \quad \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}), \qquad \boldsymbol{x} \in \mathcal{X}, \quad \boldsymbol{u} \in \mathcal{U}. \end{aligned} \tag{3}$$

As an example, the manipulation task that we analyse consists of moving an articulated object to a desired configuration. In this case the objective in (3) can be defined by:

$$||h(\boldsymbol{x}^*) - h(\boldsymbol{x}(t))|| = ||\boldsymbol{o}^* - \boldsymbol{o}||_2, \tag{4}$$

with $\boldsymbol{o}^*$ as the desired final configuration of the object. The minimization of the objective in (3) requires solving a complex optimization in a high dimensional and non-linear space. To facilitate the task, additional *surrogate objectives* are defined to introduce some bias towards the optimal solution. These objectives will appear as additional *cost terms* that penalize specific configurations. The separate terms will be presented and explained in Section IV.

[1]We define articulated objects as unactuated 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.

III. PRELIMINARIES

For the convenience of the reader, we present a brief overview of the theory supporting sampling-based control methods. For an in-depth presentation we refer the reader to [17]–[19]. This method is generally named after Model Predictive Path Integral (MPPI) control. The main algorithm can be derived by tackling the optimal control either from an information theoretic or Bayesian approach. In both cases, the central idea is to look at optimal control as the problem of finding the optimal distribution of the input. The sampling procedure can then be interpreted as a minimization of the Kullback-Leibler divergence to the optimal distribution or a gradient update which maximizes the likelihood of "good" state-input trajectories. The starting point of this approach is to consider stochastic system dynamics: $\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t) + \boldsymbol{\epsilon}(t))$, where stochasticity is expressed in the form of uncorrelated white noise $\boldsymbol{\epsilon}$. While in general stochastic dynamics are a better representation of real systems, here noise is also functional to the search of the optimal solution. Intuitively, in contrast to a deterministic system, a stochastic system has a non-zero chance to explore a region of the state space which is closer to the goal. The basic idea is to use stochasticity in the input to iteratively find better input sequences. Optimality is commonly defined with respect to the finite horizon expected cost[3]:

$$U^* = \arg\min_U \mathbb{E}_{\mathbb{Q}} \underbrace{\left[c_{\text{term}}(\boldsymbol{x}(t_f)) + \int_{t=t_0}^{t_f} c(\boldsymbol{x}, \boldsymbol{u}) dt \right]}_{J(\boldsymbol{x}, \boldsymbol{u}, t)}, \quad (5)$$

where 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 is to the goal, t_0 and t_f are the initial and final time, and $c_{\text{term}}(\boldsymbol{x}(t_f)) \in \mathbb{R}_{\geq 0}$ is the terminal cost which approximates the tail of the infinite horizon cumulative cost. Finally U is the sequence of applied inputs at every time step: $\{\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{N-1}\}$.

In the seminal work of [20] the concept of *free energy* is first introduced. This function formalizes the idea that the minimum of (5) is obtained by trading off exploration and exploitation. We define $\mathbb{P}$ as the distribution of the uncontrolled dynamics $f = f(\boldsymbol{x}, \boldsymbol{\epsilon})$ and $\mathbb{Q}$ as the distribution of the controlled dynamics $f = f(\boldsymbol{x}, \boldsymbol{u} + \boldsymbol{\epsilon})$. Then the free energy is defined as:

$$\xi(\boldsymbol{x}, t) = -\frac{1}{\lambda} \log \mathbb{E}\left[\exp(-\lambda J(\boldsymbol{x}, t))\right] \quad (6)$$

$$= \inf_{\mathbb{Q}} [\underbrace{\mathbb{E}_{\mathbb{Q}}(J(\boldsymbol{x}, t))}_{\textit{state cost}} + \underbrace{\frac{1}{\lambda} \mathbb{KL}(\mathbb{Q}||\mathbb{P}))}_{\textit{exploration cost}}]. \quad (7)$$

There exists an explicit form for the optimal distribution [19]:

$$d\mathbb{Q}^* = \frac{\exp(-\lambda J) d\mathbb{P}}{\int \exp(-\lambda J) d\mathbb{P}}, \quad (8)$$

[3] $\mathbb{E}_{\mathbb{Q}}$ is the expectation over the noise probability distribution.

from which one needs to extract the corresponding optimal policy. This is done in [21] without resorting to dynamic programming. In [21] the method consists of iteratively approximating the optimal distribution $\mathbb{Q}^*$ with a tractable known distribution $\mathbb{Q}_{U,\boldsymbol{\Sigma}}$ parametrized by its mean and variance. The procedure is to iteratively sample from $\mathbb{Q}_{U,\boldsymbol{\Sigma}}$ and change the mean of the distribution to get "closer" to the optimal by relying on the relationship in (8). This is the first example where noise is used as a means to explore the space and improve control over time. An alternative probabilistic interpretation is given in [18] and offers connections with gradient-based methods, which will be used in Section IV. An additional binary random variable $\mathcal{O}$ is first introduced, where $\mathcal{O} = 1$ denotes that a trajectory is optimal, and $\mathcal{O} = 0$ denotes that it is not [22]. In [18] each trajectory is assigned a likelihood which depends on its cumulative cost:

$$p_t(U|\mathcal{O} = 1; \boldsymbol{x}_t) = \frac{p(\mathcal{O} = 1|U; \boldsymbol{x}_t) p(U; \boldsymbol{x}_t)}{\int p(\mathcal{O} = 1|U; \boldsymbol{x}_t) p(U; \boldsymbol{x}_t)}. \quad (9)$$

Any monotonically decreasing function can be used to map costs to the space of *success likelihood*:

$$p(\mathcal{O} = 1|U; \boldsymbol{x}_t) \propto g(J) \triangleq \mathcal{J}. \quad (10)$$

The exponential utility is a common choice as in the information theoretic approach $g(J) = \exp(-\lambda J)$. The objective is then to find the posterior defined in (9). This can be done by minimizing the Kullback-Leibler divergence:

$$q^* = \arg\min_{\mathbb{Q}} \mathbb{KL}(q(U)||p(U|\mathcal{O}) \quad (11)$$

$$= \arg\min_{\mathbb{Q}} -\mathbb{E}_{\mathbb{Q}}[\log p(\mathcal{O}|U)] + \mathbb{KL}(q|p). \quad (12)$$

One can immediately recognize in (12) the same expression as in (7) where the concept of free energy and exploration-exploitation trade-off was presented. The authors propose to use stein variational gradient, an optimization method which can approximate a multimodal distribution. This method would additionally increase the computational requirements. Therefore, we keep the discussion to a single mode case. The update rule of the stein variational gradient descent method is given by $U \leftarrow U + \rho \phi^*(U)$ where ϕ^* is the optimal gradient step [23]. For the single mode case the method reduces to the optimization of the log likelihood,

$$\phi^*(U) \propto \nabla_U \log p(U|\mathcal{O}_{\boldsymbol{\tau}}; x) \quad (13)$$

$$= \nabla_U \log \mathbb{E}_q\left[\mathcal{J}\right] + \nabla_U \log \tilde{q}(U, \boldsymbol{x}_t) \quad (14)$$

$$= \nabla_U \log \mathbb{E}_q\left[\mathcal{J}\right] \quad (15)$$

$$= \frac{\mathbb{E}_q\left[\mathcal{J} \nabla_U \log \pi_U\right]}{\mathbb{E}_q\left[\mathcal{J}\right]}. \quad (16)$$

The step in (14) follows from the assumption of a uniform prior $\tilde{q}$ over the inputs. Choosing the exponential utility function and a Gaussian policy which is parametric in the mean $\pi(V) : V \sim \mathcal{N}(U, \boldsymbol{\Sigma})$ we obtain the following update

equation for the i^{th} input vector of the input sequence U:

$$\boldsymbol{u}_i = \boldsymbol{u}'_i + \rho \frac{\mathbb{E}_{\mathbb{Q}}[\exp(-\lambda J)\nabla_{\boldsymbol{u}_i} \log \pi_{\boldsymbol{u}_i}]}{\mathbb{E}_{\mathbb{Q}}[\exp(-\lambda J)]} \tag{17}$$

$$= \boldsymbol{u}'_i + \rho \boldsymbol{\Sigma}^{-1} \frac{\mathbb{E}_{\mathbb{Q}}[\exp(-\lambda J)(\boldsymbol{v}_i - \boldsymbol{u}_i)]}{\mathbb{E}_{\mathbb{Q}} \exp(-\lambda J)} \tag{18}$$

$$= \boldsymbol{u}'_i + \rho \boldsymbol{\Sigma}^{-1} \frac{\mathbb{E}_{\mathbb{Q}}[\exp(-\lambda J)\boldsymbol{\epsilon}_i]}{\mathbb{E}_{\mathbb{Q}} \exp(-\lambda J)}. \tag{19}$$

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 path integral control update rule:

$$\boldsymbol{u}_i = \boldsymbol{u}'_i + \alpha \sum_{k=0}^{K-1} \omega_k \boldsymbol{\epsilon}_i^k \tag{20}$$

$$\omega_i \approx \frac{\exp(-\lambda \tilde{J}_i)}{\sum_{k=0}^{K-1} \exp(-\lambda \tilde{J}_k)}, \tag{21}$$

where we choose the modified cumulative cost $\tilde{J}$, similar to the original derivation in [19] in order to also penalize inputs and reward sampling in regions that have not been previously explored:

$$\tilde{J} = \exp\left(-\lambda J - \lambda \sum_{i=0}^{N-1} \boldsymbol{u}_i^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{u}_i - \boldsymbol{\epsilon}_i)\right). \tag{22}$$

IV. Control Method

In this section we propose a novel stochastic method and adaptations for the control of mobile and dynamical manipulators. 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. Note that the sampling-based control is aware of low-level control if this is part of the simulated dynamics. We choose $\boldsymbol{u} = \dot{\boldsymbol{q}}$ and a low-level control of the form:

$$\boldsymbol{\tau} = \boldsymbol{K}_p(\boldsymbol{u} - \dot{\boldsymbol{q}}) + \boldsymbol{b}_r(\boldsymbol{q}, \dot{\boldsymbol{q}}), \tag{23}$$

with an appropriate choice of the positive-definite diagonal gain matrix $\boldsymbol{K}_p \in \mathbb{R}^{n_q \times n_q}$. It is advisable to set low gains in order to keep the whole system compliant and avoid high interaction forces that might arise during contact. This in turn also reduces the low-level controller bandwidth which behaves as a low-pass filter for the reference velocities, which are often non-smooth. The low-level controller continuously queries a new policy and the newest optimal input at the query time is returned. If the query time does not correspond to one of the discretized times in the optimal input sequence, the input is linearly interpolated between the time bounds of the available sequence.

A. Cost formulation

As described in Section II, the control objective is to drive the system to a desired state. The objective (3) defined in the problem formulation is equivalent to the cost in (5) when the combination of all cost terms attains its minimum at the desired state $\boldsymbol{x}^*$. 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. Input constraints are easier to handle as the non-linear dynamics can be augmented with a function that projects the sampled inputs to the feasible set $\mathcal{U}$. 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

$$\mathbb{1}[x] = \begin{cases} 1 & \text{if } x \text{ is True} \\ 0 & \text{otherwise} \end{cases}. \tag{24}$$

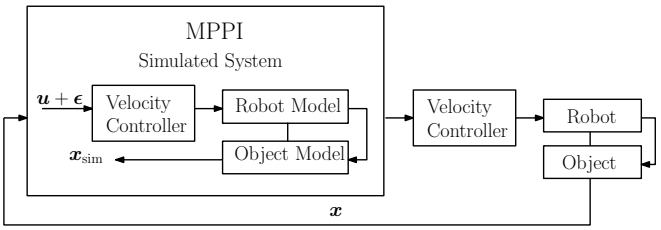


Fig. 2: The control scheme adopts a pure proportional control of joint velocities during sampling.

1) Target reaching: In the target reaching task, the goal is to bring a frame attached to the robot (generally the end-effector frame) to a desired pose. We define with the vector $\boldsymbol{p}$ the current frame pose and $\boldsymbol{p}^*$ the desired target. The *tracking cost* is computed as a weighted distance between these quantities. The distance is computed in the tangent space to $SE(3)$ using the logarithmic mapping [24]:

$$c_t(\boldsymbol{q}, \boldsymbol{W}_t) = ||\log(\boldsymbol{p} - \boldsymbol{p}^*)||^2_{\boldsymbol{W}_t}, \tag{25}$$

where $\boldsymbol{W}_t$ is a diagonal weight matrix used to penalize the translation and rotation errors. Also note that the current end-effector pose $\boldsymbol{p}$ is a function of the robot configuration $\boldsymbol{q}$, computed by evaluating the forward kinematics.

2) Collision avoidance: Let $\gamma \in \{0, 1\}$ represent an auxiliary variable which is equal to 1 when the manipulator is in contact with the environment. The value of γ can be computed by evaluating the forward kinematics for the current configuration and searching for collisions between bodies. γ is used to avoid collisions during contact-free motions of the end-effector. The *contact cost* is defined as

$$c_\gamma(\boldsymbol{x}; w_\gamma) = w_\gamma \gamma(\boldsymbol{x}), \tag{26}$$

where $w_\gamma \in \mathbb{R}_{\geq 0}$. A similar technique can be used to consider contacts between joints and penalize self-collisions.

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

$$c_j(\boldsymbol{q}; w_j, \boldsymbol{W}_{js}) = \mathbb{1}[\boldsymbol{q} > \boldsymbol{q}_{\text{upper}}](w_j + ||\boldsymbol{q}_{\text{upper}} - \boldsymbol{q})||^2_{\boldsymbol{W}_{js}}) + \mathbb{1}[\boldsymbol{q} < \boldsymbol{q}_{\text{lower}}](w_j + ||\boldsymbol{q} - \boldsymbol{q}_{\text{lower}}||^2_{\boldsymbol{W}_{js}}), \tag{27}$$

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 [21] to help the controller find its way back if poor sampling brings the system outside of

the joint position limits. Since velocities are the optimization variable of the controller, we can directly project them to the feasible set $\mathcal{U}$.

4) Arm reach: Specifically for mobile manipulators composed of a moving base and an articulated arm, we introduce an additional term that penalizes configurations where the arm's end-effector moves far from the base. This helps to select solutions where base motion is preferred over stretching the arm which can lead to singular configurations. The translation vector from the end-effector frame E to the arm base frame B is defined with the vector $\boldsymbol{t}_{BE}$. The current reach is then $r_{\text{curr}} = ||\boldsymbol{t}_{BE}||_2$. Given a maximum reach $r_{\text{max}} \in \mathbb{R}_{\geq 0}$, the *reach cost* is then defined by:

$$c_r(\boldsymbol{q}) = \mathbb{1}[r_{\text{curr}} > r_{\text{max}}](w_r + w_{rs}(r_{\text{curr}} - r_{\text{max}})^2), \quad (28)$$

with an appropriate choice of positive scalar weights w_r and w_{rs} as a constant term and a quadratic term in the violation.

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

$$c_o(\boldsymbol{o}; \boldsymbol{W}_o) = ||\boldsymbol{o} - \boldsymbol{o}^*||^2_{\boldsymbol{W}_o}. \quad (29)$$

The manipulation task consists of two phases. In a first phase the manipulator reaches the proximity of the object to be manipulated while avoiding hard collisions with the object. This step brings the robot end-effector close to an estimated contact point, allowing for a fast and successful exploration in the following manipulation phase. In the second phase, the goal is to bring the object to the desired state while keeping the end-effector close to the initial guess. The stage costs inform the controller about the different objectives in each of the described steps:

$$\begin{aligned} \text{reaching: } c(\boldsymbol{x}) =& c_t(\boldsymbol{q}; \boldsymbol{W}_t^r) + c_\gamma(\boldsymbol{o}; w_\gamma) + \\ & c_j(\boldsymbol{q}; w_j, \boldsymbol{W}_{js}) + c_r(\boldsymbol{q}; w_r, w_{rs}) \\ \text{manipulation: } c(\boldsymbol{x}) =& c_t(\boldsymbol{q}; \boldsymbol{W}_t^m) + c_o(\boldsymbol{o}; \boldsymbol{W}_o) + \\ & c_j(\boldsymbol{q}; w_j, \boldsymbol{W}_{js}) + c_r(\boldsymbol{q}; w_r, w_{rs}). \end{aligned}$$

Switching between the two phases happens when there is no continuous improvement in the stage cost. We decide to keep separate weight matrices $\boldsymbol{W}_t^r$ and $\boldsymbol{W}_t^m$ as this offers more flexibility in the cost design. As we will show in Section V, a well-engineered cost lets the controller fully exploit the contact dynamics. This approach is in contrast with previous works where manipulation generally follows a prescribed grasping state [13]. Grasping introduces a kinematic constraint that, while reducing the optimal control search space, does not allow for more flexible, contact-based behaviors to emerge such as pulling, pushing or sliding.

B. Adaptations

We now present adaptations to the main algorithm that improve efficiency and let us deploy the controller successfully on simulated and real-world whole-body manipulation tasks. The main difficulty is achieving a reasonable control rate. Monte Carlo sampling can be inefficient and naïve exploration can drive the system into local minima.

1) Momentum update: Following the Bayesian formulation, we recover the path integral update rule by taking a gradient step in the direction that minimizes the negative log-likelihood of system trajectories. Equation (19) shows that the choice of the exploration noise has the effect of tuning the gradient step size. Intuitively, a larger noise variance produces greater variability in the generated trajectories resulting in a wider exploration of the surrounding state space. On the other hand, a big step size could have detrimental effects, especially in those regions of the state space with high sensitivity to the cost function or when this is highly non-smooth. Adding a momentum term is a technique used to regularize gradient descent methods. Momentum plays the role of a short term memory and has an interesting physical interpretation: when α is small, it acts as the discretization of a damped harmonic oscillator [25]:

$$\begin{aligned} \boldsymbol{z} &= \beta \boldsymbol{z}' + \nabla f(\boldsymbol{\theta}) \\ \boldsymbol{\theta} &= \boldsymbol{\theta}' - \alpha\, \boldsymbol{z}, \end{aligned} \quad (30)$$

for a generic function f of variables $\boldsymbol{\theta}$.

Another major adaptation consists of decoupling the step size from the input variance. In this way, the explorative noise variance can be kept large, letting the algorithm explore a large region of the state space without compromising the stability of the update step by introducing dangerous oscillations in the optimized trajectory. Overall, a good choice of the parameters α and β improves performance over the naïve method and has a beneficial smoothing effect. Nevertheless, we can only access a stochastic approximation of the gradient and thus particular attention must be paid when tuning the step sizes. The update equation (20) is then modified to include momentum:

$$\boldsymbol{z}_i = \beta \boldsymbol{z}'_i + \sum_{k=0}^{K-1} \omega_k \boldsymbol{\epsilon}_i^k \quad (31)$$

$$\boldsymbol{u}_i = \boldsymbol{u}'_i + \alpha \boldsymbol{z}_i, \quad (32)$$

where the sign change follows from doing gradient ascent for the likelihood function. Note that the the original update is recovered by setting $\alpha = 1$ and $\beta = 0$.

2) Receding Horizon Control: Model-based predictive control rarely achieves the rate of common low-level controllers (around 100Hz or higher depending on the system) because of the intensive optimization performed at every iteration step. Nevertheless, the predicted horizon can be used in open loop by a cascaded trajectory tracking controller before the new optimization step has completed. The rate of the low-level controller is generally higher than the upstream stochastic controller. Therefore the stochastic controller keeps an internal copy of the previously optimized trajectory. Then, the low-level controller executes in open loop the input sequence. The sequence is swapped by the controller once a new one is available. Thus, the two blocks are decoupled.

At the beginning of a new optimization all the stored rollouts are shifted back by the real-time number of steps elapsed since the last optimization. A subset of the best

rollouts (assuming they are ordered) are kept to warm start the next iteration. For these, the actual deviation from the optimal input needs to be recomputed.

The solutions provided by the sampling-based controller are not sufficiently smooth for a practical application, especially with a big step size and no momentum. Poor smoothness limits transfer to the real hardware that, because of the limited control bandwidth, could quickly wear out. In order to filter the solution without introducing additional delays, we apply a moving window Savitzky-Golay filter [26]. In contrast to previous approaches, we keep a history of previously applied inputs in order to avoid filtering discontinuities and unintentional delays during near-real-time operations.

3) Adaptive temperature: The coefficient λ in (20) determines how "aggressive" the weighting is between different trajectories. 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 equivalently good can be assigned very different weights. We instead propose to adopt the same technique as in [27] where the parameter λ can be automatically optimized to maximally discriminate between the experienced trajectories. The modified exponential utility is then defined by,

$$\exp(-\lambda J) = \exp\left(-h\frac{J - J_{\min}}{J_{\max} - J_{\min}}\right). \tag{33}$$

V. Experimental Results

In this section we present an evaluation of the proposed method both in simulation and on hardware. As the framework is generic, additionally to the dynamical system described in Section II, we design a challenging scenario where the goal is to control a 7-DOF manipulator mounted on a non-holonomic base. For this system, the control inputs are the joints velocities for the arm, and the forward and rotational velocities for the base. The equations of motion for the kinematic manipulator are,

$$\begin{aligned} \dot{p}_x &= \cos(\theta) v_x \\ \dot{p}_y &= \sin(\theta) v_x \\ \dot{\theta} &= v_\theta \\ \dot{\boldsymbol{q}}_{arm} &= \tilde{\boldsymbol{u}}, \end{aligned} \tag{34}$$

where p_x, p_y, θ are the 2D position and yaw angle of the base, while the input vector is denoted by $\boldsymbol{u}^\top = [v_x, v_y, v_\theta, \tilde{\boldsymbol{u}}^\top]^\top$. We deploy the kinematic model for a target-reaching task where several desired end-effector target poses are set in sequence as illustrated in Fig. 3. Notice that this toy example could also be deployed as a kinematic planner on a real system. The dynamic model as described by (2) is then tested on complex manipulation tasks demonstrating the real-time capabilities of the algorithm. These consist of maneuvering different articulated objects. Over all of our simulation trials, we never experience failure, which means that the algorithm always succeeds at performing the task. The articulated objects in the task are a *shelf*, *dishwasher*, *microwave* and *drawer* as shown in Fig. 4. They differ in type and orientation of the joint. The *shelf* and *microwave* have a vertical revolute joint while the *dishwasher* has a horizontal revolute joint. Finally, the *drawer* has a horizontal prismatic joint. We evaluate the adaptations to the main algorithm and provide observations about tuning the simulation and the algorithm parameters such as the sampling budget.

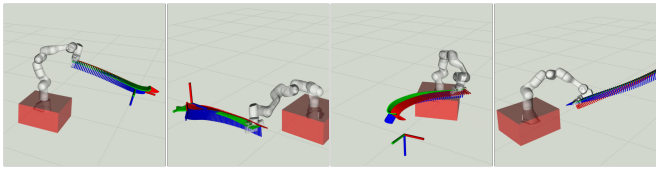


Fig. 3: Excerpts of the target sequence used for algorithm evaluation. The optimal planned end-effector path is also visualized.

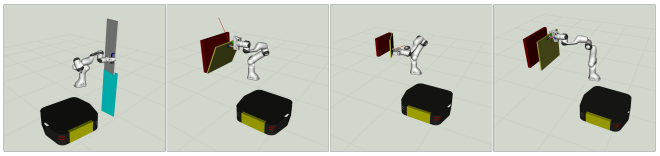


Fig. 4: The four articulated objects used in our simulation evaluations. From left to right: shelf, dishwasher, microwave, drawer.

A. Momentum update

We first test the novel *momentum update* on each of the tasks in simulation. In this experiment we compare against the naïve algorithm which we recover when choosing $\alpha = 1$ and $\beta = 0$. As we can see in Fig. 5, adding momentum can often be beneficial.

B. Tuning the sampling budget

We also investigated performance as a function of the sampling budget. It is generally argued that more samples provide a better solution [21]. Nevertheless, in this work we reinterpret the update equation (20) as a noisy stochastic estimate of the gradient in (13). The update then corresponds to a mini-batch stochastic gradient descent. In fact, mini-batches consisting of different rollouts average out noise but the estimate variance is still high enough to escape local minima. Furthermore, we observe that there is no considerable performance gain above a certain number of sampled rollouts. Instead, sampling more trajectories has the detrimental effect of increasing the computing time and reducing the control rate. This observation is crucial when optimizing for algorithm efficiency and is visualized in Fig. 6.

C. Tuning the rollout simulation

A correct tuning of the simulator is of vital importance as this is the core component of the controller and can improve performance in terms of control rate and quality of the solution. 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. 7. 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. The original collision mesh, besides being

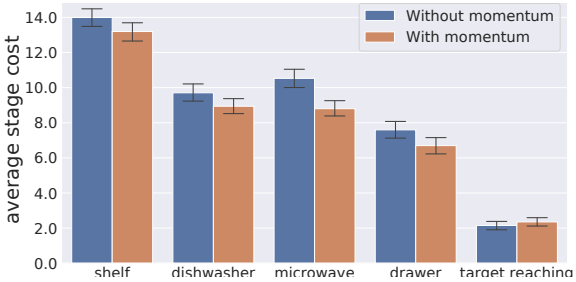


Fig. 5: We evaluate the momentum update on two challenging tasks: end effector target-reaching for a non-holonomic kineamatic system and object manipulation for a full dynamic manipulator. Statistics are taken over 3 runs. α and β are set to 0.1 and 0.99 respectively when momentum is used.

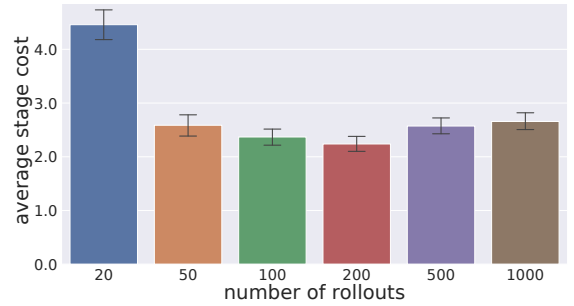


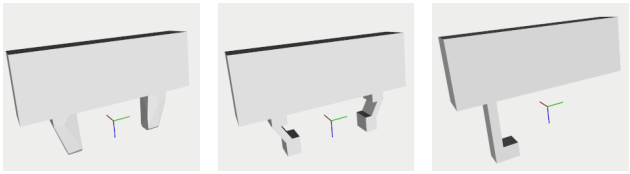
Fig. 6: Average cost as a function of the number of sampled rollouts for the target reaching task of a mobile manipulator on a non-holonomic base. The trend shows that performance does not necessarily increase with more samples.

inaccurate, slows down collision checking. A simplified yet accurate model consisting of primitive shapes produces a significant speed-up as we can see in Fig. 8. Some additional statistics are also presented in Table I.

	Min Rate [Hz]	Max Rate [Hz]	Average Rate[Hz]
Original mesh	10	306	145
Two fingers	**49**	321	161
Hook finger	44	**327**	**177**

TABLE I: Control rate statistics when using different collision shapes for the door opening task.

Another important aspect is the discrepancy between the simulator and the real physical model. In fact, we have experienced numerous initial failures when transferring the controller to the real hardware that were caused by model mismatch. In particular, the controller would try to open the door by "scratching" the handle with the finger tip. This approach could be immediately related to the fact that the simulator assumed a high friction coefficient between the two materials. In practice friction is hard to measure and depends on the contact patch between surfaces. This is extremely challenging to transfer from simulation to hardware. Instead kinematic constraints between contact points depends on the system geometry which can be accurately measured. 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. We corrected the friction to a much lower value (0.01), and managed afterward to produce consistent successful trials on the real platform.



(a) (b) (c)

Fig. 7: This figure shows three different collision shapes for the robot end-effector: (a) original mesh, (b) simplified collision shape, (c) modified hook design. Original collision meshes are often approximated by convex hulls which are inaccurate and more complex representations.

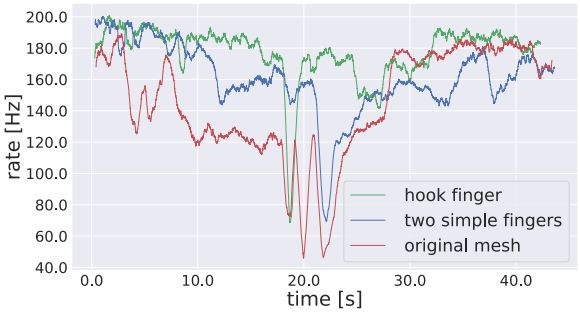


Fig. 8: We compare the controller rate for the door opening task when deploying different collision shapes. A simplified, yet accurate model significantly increases the controller rate, especially during the contact phase, visible as a drop in the frequency. Each run corresponds to 20 rollouts, 1s horizon and a time step of 0.015s.

D. Real world experiments

Our RoyalPanda test platform consists of a holonomic mobile base equipped with a 7-DOF manipulator. The robot's wrist mounts a custom set of fingers as shown in Fig. 9. This hardware adaptation simplifies the problem without limiting the capabilities of the platform. We run the presented algorithm 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. We filter the base input with a window of 50 samples and a third order polynomial. The joint velocities are filtered with a window of 10 samples and a first order polynomial. The input variance $\mathbf{\Sigma}$ is set to $0.25I$. We keep a value of $\alpha = 0.1$ and $\beta = 0.8$ for all hardware experiments. The target velocity commands are tracked by a PI controller which converts these into motor torques. 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.

The experiment goal is to demonstrate that the algorithm can be deployed on a real platform at high control rates. For this purpose, we perform a door opening experiment. The door and the robot base are tracked via a VICON system, eliminating the need for precise state estimation. We plan to remove this limitation in future work. In order to qualitatively evaluate the algorithm's replanning capabilities, we disturb the manipulator during the opening phase releasing the contact between the handle and the finger. As we can see in the accompanying video, the controller is able to replan a feasible trajectory to the handle and successfully perform the task. In Fig. 10 the magnitudes of the base and end-

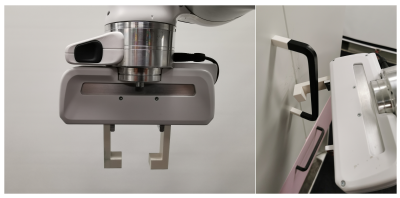
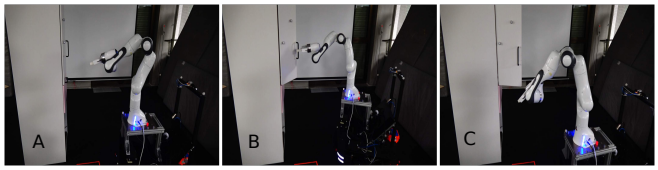
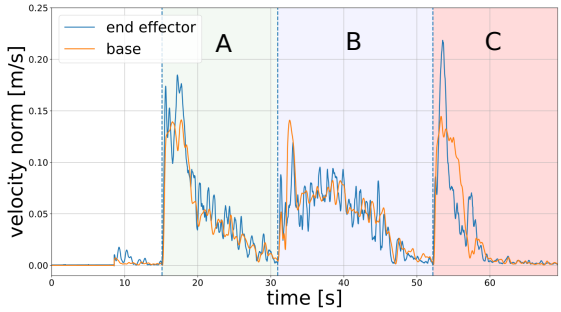


Fig. 9: 3D printed fingers used during the experiment. The presence of the tooth encourages the controller to exploit contact between the finger and the handle.



(a) Extract of the manipulation sequence. In A the robot reaches an estimated target region near the handle. In B the door is opened. In C the end-effector moves back to the starting pose.



(b) Velocity contributions of the base and of the end-effector with respect to the base frame. These results show that whole-body coordination occurs at all the times of the manipulation task.

Fig. 10: Whole-body door opening with a mobile manipulator.

effector velocities are plotted to visualize the whole-body coordination throughout the manipulation task.

VI. Conclusions

To the best of the authors' knowledge, this is the first time that the path integral framework has been successfully applied on a 10-DOF mobile manipulator. We have shown that the algorithm can be applied to control a complex dynamical system for contact-based tasks. Real-time control is achieved without the need for massive parallel computation but rather our method exploits fast, readily-available CPU simulators and several algorithmic adaptations that let us successfully transfer the method onto a real platform. We introduce and evaluate *momentum update* to improve convergence and stability of the optimized input sequence. Lastly, we provide some practical observations and a open source *robot-agnostic* implementation.

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] M. Brunner, K. Bodie, M. Kamel, M. Pantic, W. Zhang, J. Nieto, and R. Siegwart, "Trajectory tracking nonlinear model predictive control for an overactuated mav," in *2020 IEEE Int. Conf. on Robot. and Autom.*, 2020, pp. 5342–5348.

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

[18] A. Lambert, A. Fishman, D. Fox, B. Boots, and F. Ramos, "Stein variational model predictive control," in *Conf. on Robot Learn.*, 2020.

[19] E. A. Theodorou, "Nonlinear stochastic control and information theoretic dualities: Connections, interdependencies and thermodynamic interpretations," *Entropy*, vol. 17, no. 5, pp. 3352–3375, 2015.

[20] H. J. Kappen, "Path integrals and symmetry breaking for optimal control theory," *J. of Statistical Mechanics: Theory and Experiment*, vol. 2005, no. 11, p. P11011, 2005.

[21] 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.

[22] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," *arXiv preprint arXiv:1805.00909*, 2018.

[23] Q. Liu and D. Wang, "Stein variational gradient descent: a general purpose Bayesian inference algorithm," in *Proc. of the 30th Int. Conf. on Neural Inf. Process. Syst.*, 2016, pp. 2378–2386.

[24] J.-L. Blanco, "A tutorial on SE(3) transformation parameterizations and on-manifold optimization," University of Malaga, Tech. Rep., 2013.

[25] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.

[26] 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.

[27] 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.